High Performance and Low Power On-Die Interconnect Fabrics

by

Sudhir Kumar Satpathy

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering)
in The University of Michigan
2012

Doctoral Committee:

Professor David Blaauw, Chair
Professor Trevor N. Mudge
Professor Dennis Michael Sylvester
Associate Professor Kevin Patrick Pipe
Assistant Professor Zhengya Zhang
Senior Principal Engineer Ram K. Krishnamurthy, Intel Corporation

To my parents and sister, Kalpana for their love and support

ACKNOWLEDGEMENTS

First and foremost, I offer my sincerest gratitude to my advisor, Prof. David Blaauw for his guidance and support in all my research endeavours. I would like to thank him for the extreme degree of freedom that he has given me while conducting research, that has made my experience at the University of Michigan so pleasurable. I would also like to thank all the other committee members, Prof. Dennis Sylvester, Prof. Trevor Mudge, Prof. Zhengya Zhang, Prof. Kevin Pipe and Dr. Ram Krishnamurthy for providing valuable feedback and support.

I would like to thank all students of Michigan Integrated Cicruits Laboratory for sharing their time and expertise. Thanks to all students from Trev's research group for their help in the projects that contribute to this dissertation. Thanks to ARM Ltd. for funding my research during all these years. Thanks to Intel's Circuits Research Lab (CRL) for giving me the opportunity to spend a wonderful summer in Hillsboro while working with some of the best VLSI researchers. Thanks to Himanshu and Mark for their excellent mentorship and guidance during my internship at CRL in summer of 2011.

I would like to thank my parents, my younger sister Kalpana, friends and relatives for their support and encouragament. Last but not least, I want to thank Almighty for blessing me with the mental aptitude and physical ability to achieve my goals.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

ABSTRACT

High Performance and Low Power On-Die Interconnect Fabrics

by

Sudhir Kumar Satpathy

Chair: David Blaauw

Increasing power density with technology scaling has caused stagnation in operating frequency of modern day microprocessors. This has led designers to prefer multi-core architectures over complex monolithic processors to keep up with the demand for rising computing throughput. Although processing units are getting smaller and simpler, the dramatic rise of their count on a single die has made the fabric that connects these processing units increasingly complex. These interconnect fabrics have become a bottleneck in improving overall system efficiency. As a result, the design paradigm for multi-core chips is gradually shifting from a core-centric architecture towards an interconnect-centric architecture, where system efficiency is limited by the fabric rather than the processing ability of any individual core. This dissertation introduces three novel and synergistic circuit techniques to improve scalability of switch fabrics to make on-die integration of hundreds to thousands of cores feasible. 1) A matrix topology is proposed for designing a fully connected switch fabric that reuses output buses for programming, and stores shuffle configurations at cross points. This significantly reduces routing congestion, lowers area/power, and improves performance. Silicon measurements demonstrate 47% energy savings in a 64-lane SIMD

processor fabricated in 65nm CMOS over a conventional implementation. 2) A novel approach to handle high radix arbitration along with data routing is proposed. It optimally uses existing cross-bar interconnect resources without requiring any additional overhead. Bandwidth exceeding 2Tb/s is recorded in a test prototype fabricated in 65nm. 3) Building on the later, a new circuit topology to manage and update priority adaptively within the switch fabric without incurring additional delay or area is then proposed. Several assist circuit techniques, such as a thyristor based sense amplifier and self regenerating bi-directional repeaters are proposed for high speed energy efficient signaling to and from the switch fabric to improve overall routing efficiency. Using these techniques a 64×64 switch fabric with 128b data bus fabricated in 45nm achieves a throughput of 4.5Tb/s at single cycle latency while operating at 559MHz.

CHAPTER I

Introduction

1.1 Technology scaling and the multi-core era

Semiconductor computing technology has advanced by leaps and bounds since its inception because of ever reducing transistor size as predicted by Moores law [1]. With on die computing resources doubling every two years and aided by transistors that switch faster, foundries continually delivered single core processors that were faster, smaller and more energy efficient over their predecessors. Even though transistor count continues to grow steadily, rising power density with technology scaling has recently caused core operating frequency to stay stagnant. This is shown in Fig. 1.1 and Fig. 1.2 which list the transistor count, and operating frequency of some commercial microprocessors reported in the last two decades.

This has led designers switch over to multi-core architectures from complex monolithic processors to address the demand for higher computing throughput. High end servers [3][4][5], gigabit Ethernet routers [6], multiprocessor NoCs, and multimedia processors [7][8] now serve workloads that process terabytes of data flow every second. Such compute capability is beyond the reach of todays single core chips that operate only within a small power budget. Even many medium throughput applications now prefer multi-core architectures over a single core implementation for better energy efficiency and reliability management [9]. With transistors getting cheaper and faster,

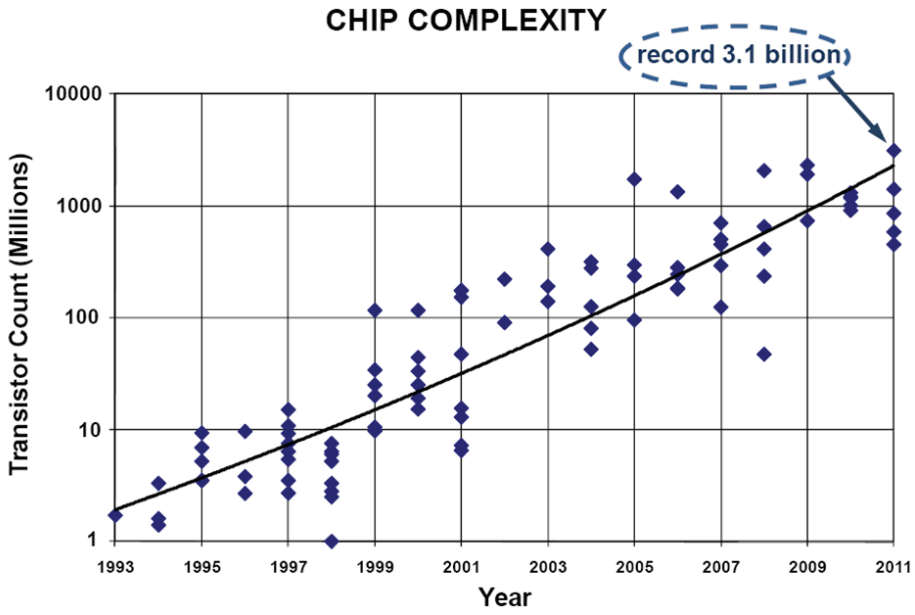

Figure 1.1: Increasing chip complexity trend in last two decades [2]

the core count in multi-processor systems has been steadily increasing [2] as shown in Fig. 1.3. Some notable multi-core chips that have been fabricated in recent years are intel's 80 Tile Teraflop compute platform [10], MIT's 64 Tile RAW Processor [11], intel's 48-Core Single-Chip Cloud computer [12], 167-Core AsAP programmable array processor chip [13], and Tilera's 64-Core TILEPro64 chip [14].

High radix, high bandwidth, and low latency switch fabrics are key enablers for manycore computers. These systems invariably need a communication network to permute data among processing and storage units in the chip. Although processing units are getting smaller and simpler taking full advantage of technology scaling, the dramatic rise in their count on a single die has resulted in the growing complexity of the interconnect fabric. The fabric size usually grows quadratically with the number of IPs that it links [15][16]. The average latency and energy spent for sending information across these fabrics increases more than linearly. This partly nullifies the throughput gains achieved from multi-core computing. It is now the case that switch fabrics are a bottleneck in improving overall system efficiency. As a result, the design paradigm for multi-core chips has shifted from a core centric architecture towards an

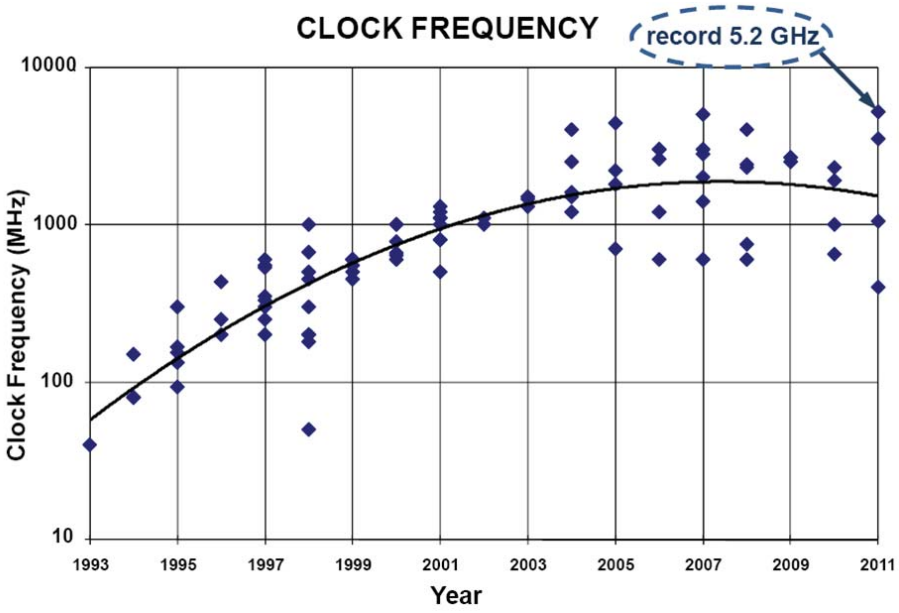

Figure 1.2: Processor frequency trend in last two decades [2]

interconnect centric architecture. In such a design space overall system efficiency is limited by the bandwidth of the interconnect fabric rather than the processing ability of any individual core.

1.2 Generic switch fabric

A generic switch fabric comprises of three key modules as shown in Fig. 1.4. 1) A data routing module to transfer information among different IPs connected to the fabric. This could be a single or a collection of shared buses for systems where processing units rarely communicate [17][18][19], or could be a fully connected crossbar for systems where performance is constrained by the fabric's bandwidth [20]. 2) An arbiter that receives requests from processing units and configures the fabric to ensure that data sent from a source reaches the appropriate destination. 3) A priority management module that monitors traffic flow pattern within the fabric and assists the arbiter to ensure fairness in resource allocation.

Traditionally these modules have been looked into in great detail independently

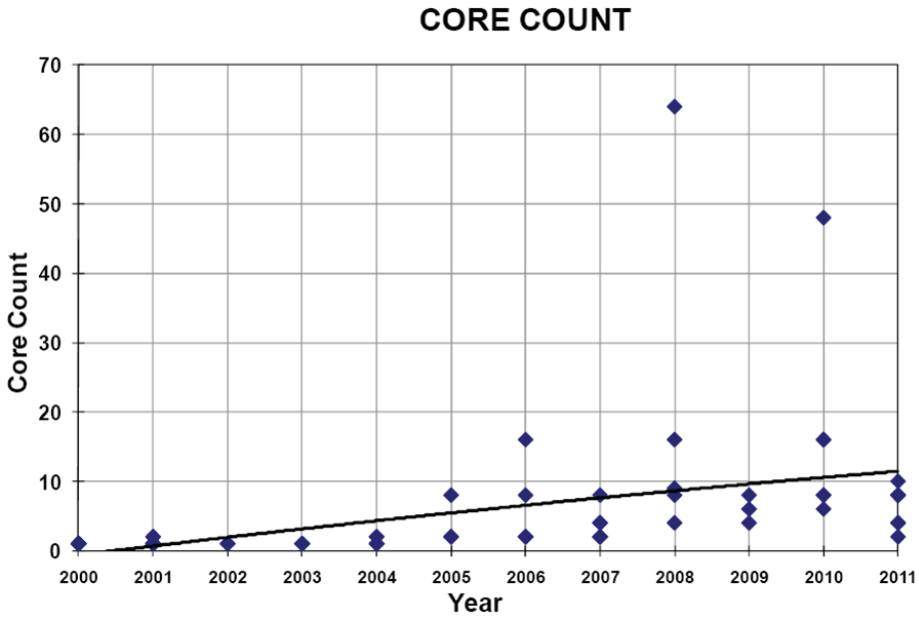

Figure 1.3: Processor core count trend in last two decades [2]

[21][22][23][24]. Many variants of the data routing hardware such as a shared bus, fat tree, banyan network, benes network, clos network, flattened butterfly network, torus fabric, fully connected crossbar etc. have been proposed in the literature [25][26][27][28]. Some recent implementations use hierarchical interconnect fabrics that are made from multiple switch topologies [29][30][31] to take advantage of varying traffic patterns at different levels of design hierarchy. Similarly, various implementations of the arbiter and priorty management module are also available in the literature [32][33][34]. Even though a true least recently granted (LRG) arbitration policy guarantees fairness and deadlock free network operation, the complexity of its implementation in high radix networks limits its usage. Hence, designers have often resorted to simple techniques such as round robin or random selection protocols for easy implemetation [35][36][37]. Though such protocols work well over a large set of applications, at times they can lead to starvation in certain parts of the network, and hence are not recommended for systems that deal with time critical workloads.

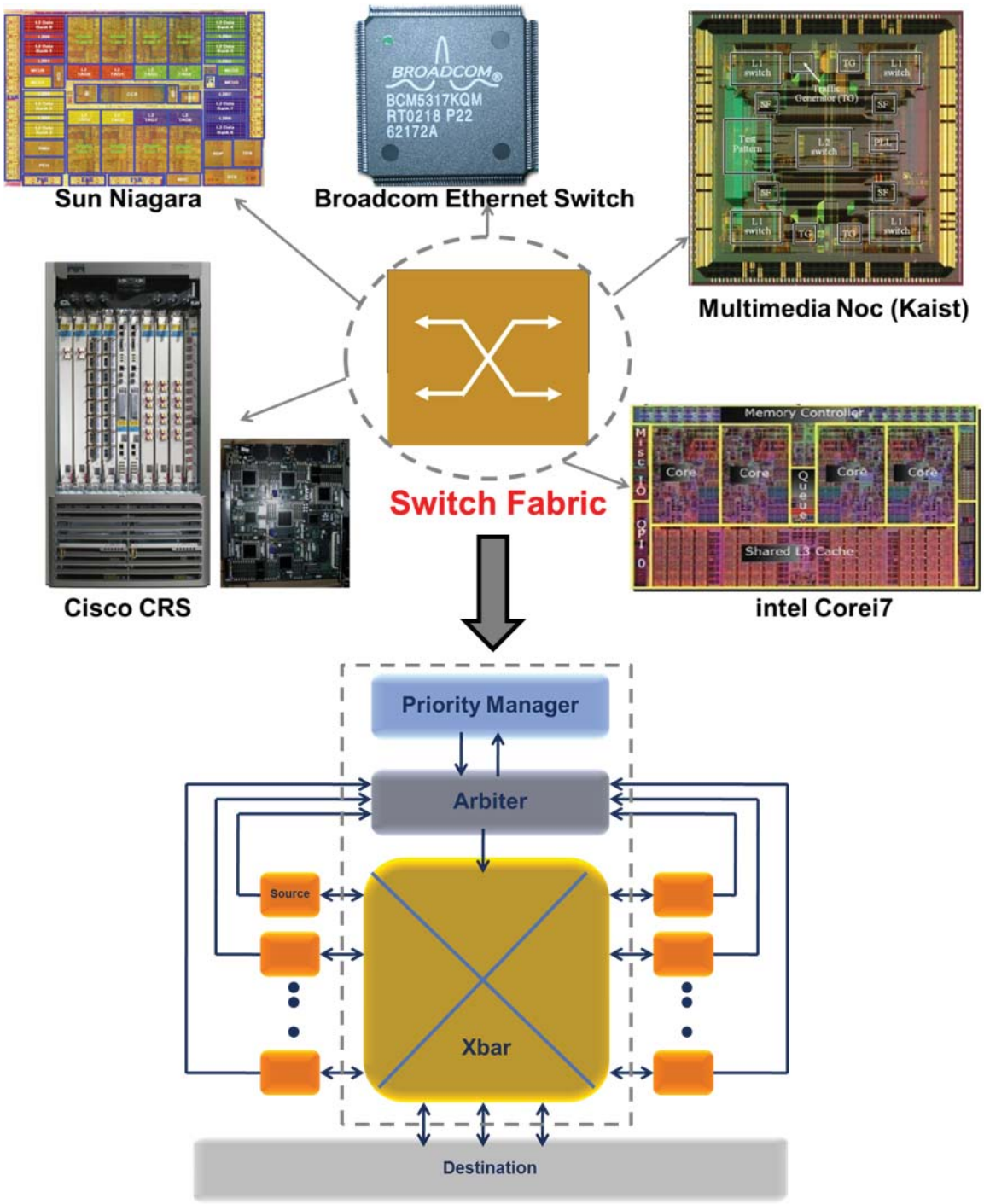

Figure 1.4: Generic switch fabric topology

1.3 Challenges for scalability

1.3.1 Design challenges

As shown in Fig. 1.4, the overall efficiency of the switch fabric relies on how efficiently each of the independent modules function and how seamlessly they communicate among one another. With a growing number of input and output ports, each module gets physically bigger and hence farther from the others. Beyond a size, the latency and energy overhead due to communication between these modules starts limiting overall fabric efficiency. Existing circuit techniques to build high radix switch fabrics rely on assembling together smaller switches that are usually 5×5 in dimension [43][44][45][46][47][48]. This approach has certain limitations: 1) The elementary switches are built using multiplexers that select bits rather than buses. Hence, as buses get wider, routing at the fabric input ports involves a lot of swizzling among the wires incurring additional area penalty. 2) A switch with ports far exceeding 5 would require multiple of these switches to be connected in stages. Data has to traverse through multiple stages to reach its destination thereby increasing latency and energy dissipation. They would also require additional data storage elements in the data routing path for higher throughput resulting in futher latency and power overhead. 3) As shown in Fig. 1.4, each source sends requests to the arbiter before it could access a destination, and eventually the arbiter sends an acknowledgement back to the source after setting up the routing path. Configuring all the switches along the routing path incurs latency. In sytems that have well defined communication patterns and usually operate on massive sets of data, the latency and energy cost for configuring the fabric can be amortized by setting up the routing path ahead of time or by sending multiple chunks of data once the path is established. However, in generic multiprocessor systems most traffic patterns are not pre-defined. Hence, the fabric configuration cost becomes a bottleneck. 4) A non-blocking switch sup-

ports all possible permutations (1-1 mappings of input to output ports as shown in Fig. 1.5) and hence can guarantee starvation free communication for all applications. However many applications particularly in the area of signal processing (like FFT, LDPC, Color space conversion etc.) can be sped up significantly by incorporating multicast and broadcast features in the switch fabric [49]. Multicasts are mappings that allow individual inputs to connect to multiple outputs, however, no output may be connected to more than 1 input. Broadcast is a special form of multicast where a single input is connected to all the outputs. A multiplexer based fabric built in multiple stages does not naturally multicast and would require significantly more logic to incorporate these features.

Analysis of different arbitration policies in the literature show that arbitration schemes have a noticeable impact on throughput and fairness of interconnection networks [38][39][40][41]. Some arbitration policies like the greedy allocation policy tend to maximize network throughput at the cost of quality of service. At the other extreme, some fancy schemes like probabilistic distance [60] based arbitration can guarantee quality of service at the price of more complex logic and hence additional arbitration latency and power overhead. Hence, adaptive and hybrid resource allocation schemes are preferred in general over static schemes because of their ability to mitigate congestion and hot spot in networks by re-directing traffic flow.

1.3.2 Technology scaling imposed challenges

Apart from the above mentioned design challenges, advancement in process technologies has posed new challenges for on die switch fabrics. Interconnect pitch has not been scaling as aggresively as transistor size for sub 65nm process nodes. Traditional switch fabric topologies are heavily wiring limited with logic utilization as low as 60% at high radices. The size of the fabric, and hence the latency and energy dissipation are primarily determined by the wiring pitch. So smaller technology nodes that do not

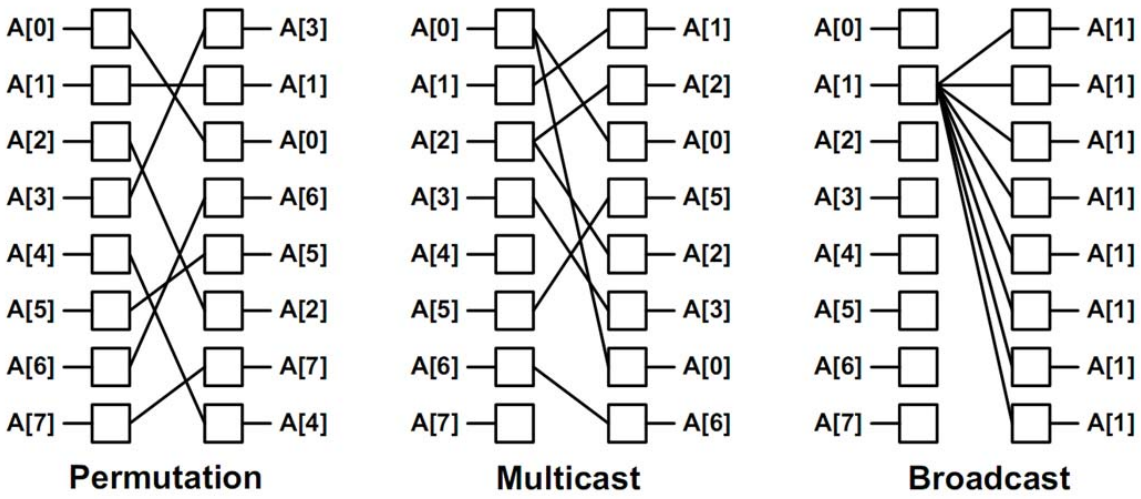

Figure 1.5: Traffic patterns in a switch fabric

offer significant reduction in interconnecr pitch, resistance and capacitance, cannot be leveraged to their full potential in designs that are fabric limited. Besides, with logic area shrinking and power density increasing with technology scaling, a higher percentage of available on die routing resources now need to be devoted towards power distribution. This leaves fewer tracks available for signal routing. Again, with coupling and capacitive crosstalk getting worse, designs in smaller technology nodes require higher spacing to maintain similar signal integrity. This leaves even fewer usable routing tracks. These problems aggravate the already mentioned limitations imposed by traditional circuit techniques to build switch fabrics.

1.4 Dissertation contribution

This research proposes architectures and circuit techniques for improving overall network efficiency and scalability. Prior research contributions in this area pertain towards improving specific parts/functionalities within the fabric, thereby reducing their efficacy in addresing challenges poised by the fabric as a whole. By narrowing down the region of interest, most prior work have been unable to leverage design slack

available in one module of the network to optimize other parts. In this dissertation we introduce and demonstrate in silicon unique fabric architectures aided by novel circuit techniques that co-optimize data routing, arbitration and resource allocation simultaneously to improve overall network scalability. Much of the success of our approaches come from our ability to integrate multiple fabric functionalities onto the same hardware with negligible reconfiguration overhead, thereby making optimum use of available on-die silicon real estate and routing tracks. Some key accomplishments of this research are :

1) The switch fabrics fabricated in this research are some of the densest fabrics reported in literature.

2) The switch fabrics use unique circuit techniques that make high radix implementations feasible. In contrast with most state-of-the-art fabrics that are built from 5×5 switches, we have built fabrics using single stage switches with dimensions 128×128 (16b data bus), 32×32 (64b data bus), and 64×64 (128b data bus).

3) Contemporary high radix switch fabrics achieve high throughput (in the order of Tb/s) by trading off latency. In contrast, fabrics reported in this thesis have achieved similar throughput at substantially lower communication latency. The proposed fabrics require one data transfer cycle for swicth configuration and resource allocation irrespective of their radices.

4) These fabrics can be used as a drop in replacement for other fabrics in multiprocessor chips. We have demonstrated up to 47% energy savings in SIMD (single instruction multiple data) processors and 1.6× reduction in conflicting memory latency in a 64-core system while using these proposed fabrics in place of conventional ones.

We demonstrated the circuit techiques proposed in this thesis by buiding some silicon prototypes as listed below.

1.4.1 XRAM: An SRAM inspired swizzle network

XRAM [50][52] is a circuit switched swizzle network. It uses an SRAM-based approach producing a compact fabric footprint that scales well with network dimensions while supporting all permutations and multicasts. Capable of storing multiple shuffle configurations and aided by a novel sense-amp for robust bit-line evaluation, a 128×128 XRAM with 16b data bus fabricated in 65nm bulk CMOS achieves a band-width exceeding 1Tbit/s. It enables a 64-lane SIMD engine operating at 0.72V to save 46.8% energy over an iso-throughput conventional 16-lane implementation operating at 1.1V.

1.4.2 SWIFT: Swizzle Interconnect Fabric Topology

SWIFT [51] is a 32×32 (64b data bus) self-arbitrating switch fabric. It achieves a bandwidth of 2.1Tb/s with single cycle arbitration and data transfer latency in 65nm bulk CMOS while operating at 1026MHz at 1.2V. SWIFT co-optimizes arbiter and crossbar logic using a unique fabric architecture that integrates conflict resolution with data routing to optimally use logic and interconnect resources. It spans 0.35mm^2, achieves an efficiency of 7.39Tbps/W, and operates down to 530mV.

1.4.3 SSN: Swizzle Switch Network with self-updating least recently granted priority and quality of service arbitration

SSN [42] is a 64×64 (128b data bus) switch fabric with a throughput of 4.5Tb/s at 3.4Tb/s/W energy efficiency. It spans 4.06mm^2 in 45nm SOI CMOS and achieves a peak efficiency of 7.4Tb/s/W at 0.6V. It features a single cycle least recently granted arbitration technique that re-uses data buses and switching logic. It also integrates 4-level message based priority arbitration for quality of service and unique bi-directional bit-line repeaters for improved scalability. The novel single cycle LRG implementation in SSN reduces worst-case cache access latency by 1.83× and 2.03× on average over

round robin and random arbitration schemes for SPLASH 2 multiprocessor benchmarks in a 64-core 64-cache system.

1.4.4 TABS: Thyristor Assisted Bi-directional Signalling

TABS is a thyristor-assisted standard cell compatible self-timed bi-directional repeater with no configuration overhead. It enables 8mm interconnects to achieve 37% higher speed at 20% lower energy over conventional repeaters in 65nm CMOS at 1.0V. In TABS absence of configuration logic removes the need for clocking, yielding up to 14× higher energy efficiency at very low data switching activity over conventional repeaters.

1.5 Dissertation outline

The remainder of this dissertation is organized as follows. In chapter 2, we address the limitations of traditional circuit techniques in designing high radix data permutation networks and propose a novel fabric architecture called XRAM. We also introduce the technique of caching frequently used shuffle patterns within the permutation network and demonstrate its benefits in a SIMD test prototype. In chapter 3, we address the challenges involved in designing high radix arbiters by introducing our second generation fabric called SWIFT. Building on this, in chapter 4 we introduce the third generation switch fabric called SSN that leverages unique circuit techniques to guarantee fast deadlock free routing by using the least recently used resource allocation policy. All the above mentioned architectures heavily rely on reusing interconnect resources to accomplish different fabric functionalities. The wires in these fabrics run for millimeters and are driven by multiple drivers at different locations. To optimize bi-directional signaling on such wires, in chapter 5 we propose TABS which is a standard cell replacement for conventional bi-directional repeaters. TABS uses a unique circuit tehnique that obviates the need for any direction configuration logic

and clock, thereby improving speed as well as energy efficiency. Chapter 6 conludes this dissertation by presenting directions for future research in this area.

CHAPTER II

XRAM : An SRAM Inspired Swizzle Network

2.1 Introduction

The permutation network is one of the key building blocks in the interconnect fabric. A fully connected matrix type crossbar can accomplish all possible data permutations including multicast and broadcast. However, its control overhead grows quadratically with the number of inputs and outputs, thereby limiting its scalability. In this chapter we introduce a new permutation network called XRAM to optimize this particular aspect of the switch fabric as shown in Fig. 2.1. XRAM is a novel circuit switched interconnect fabric. Fig. 2.2 shows the top level diagram of XRAM fabric. It uses an SRAM-based approach that results in a compact silicon footprint that scales well with network dimensions. It supports all permutations and multicasts. Capable of storing multiple shuffle configurations and aided by a novel sense-amp for robust bit-line evaluation, a 128×128 XRAM with 16b data bus fabricated in 65nm CMOS achieves a bandwidth exceeding 1Tbit/s, enabling a 64-lane SIMD engine operating at 0.72V to save 46.8% energy over an iso-throughput conventional 16-lane implementation operating at 1.1V.

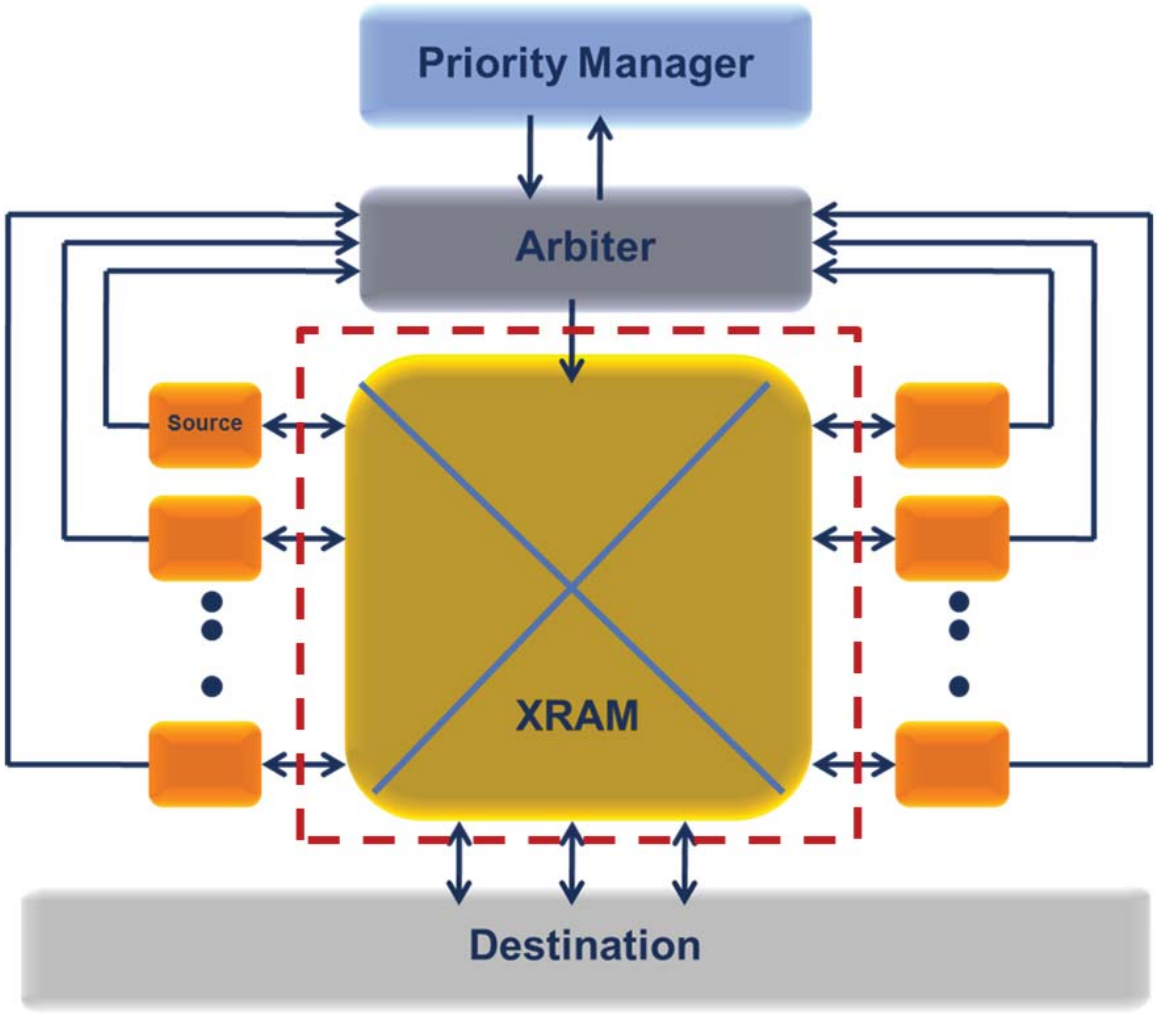

Figure 2.1: XRAM as the permutation network

2.2 System overview

The test prototype is a 5 stage 64 lane vector processor with a dedicated pipeline stage to shuffle operands before computation as shown in Fig. 2.3. The shuffle stage uses a 128×128 XRAM as a non-blocking fully connected fabric to re-arrange vector operands read from a 16 entry, 1024 bit (64 chunks of 16 bit) register file before feeding it into the execution stage. The execution stage consists of a bank of 64 processing units. Each processing unit is comprised of a 16 bit multiplier and an ALU. Every cycle a pair of 1024 bit operands is shuffled in accordance with a pre-

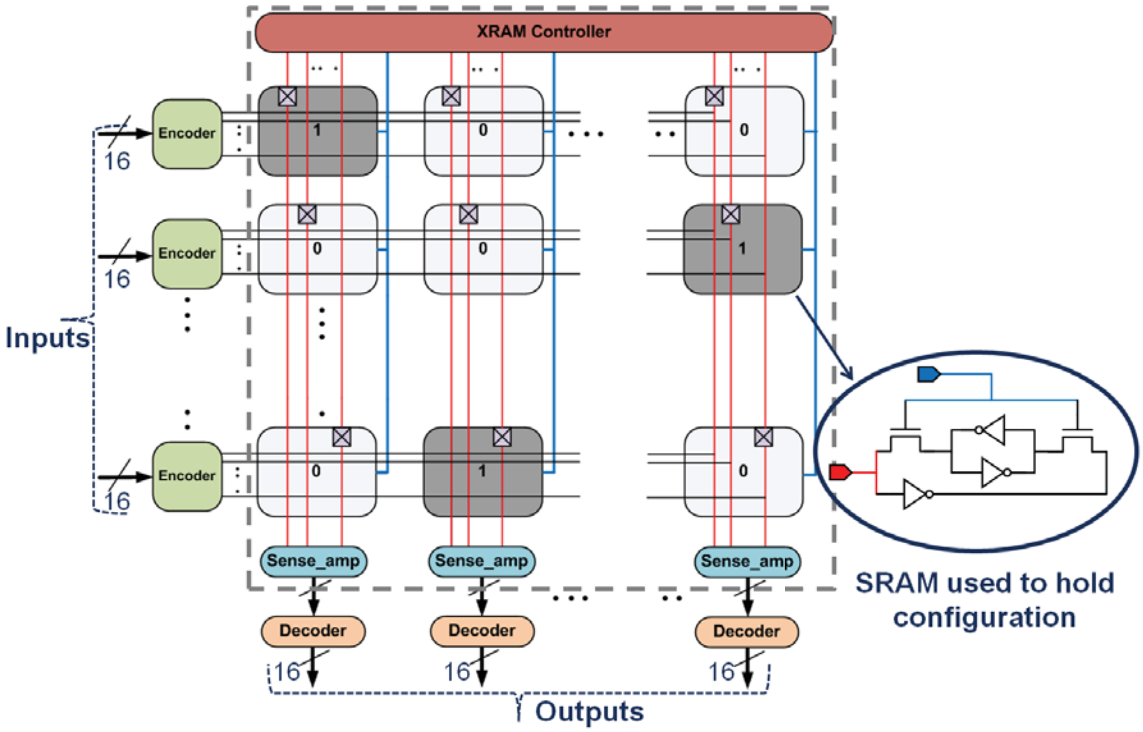

Figure 2.2: XRAM fabric topology

defined pattern configured into the XRAM. A 16 bit scalar pipeline acts as a host processor to configure the XRAM and manage permutation patterns for the vector data path.

2.3 XRAM architecture

XRAM is a matrix crossbar that leverages some of the circuit techniques used in SRAM arrays for improving area and performance. Fig. 2.2 shows a top level diagram of XRAM. The input buses span the width while the output buses run perpendicular to them, creating an array of cross points. Each cross point contains a 6T SRAM bit cell. The state of the SRAM bit cell at a cross point determines whether or not input data is passed onto the output bus at the cross point. Along a column only one bit cell is programmed to store logic high and create a connection to an input. Matrix type crossbars incur a huge area overhead because of quadratically increasing

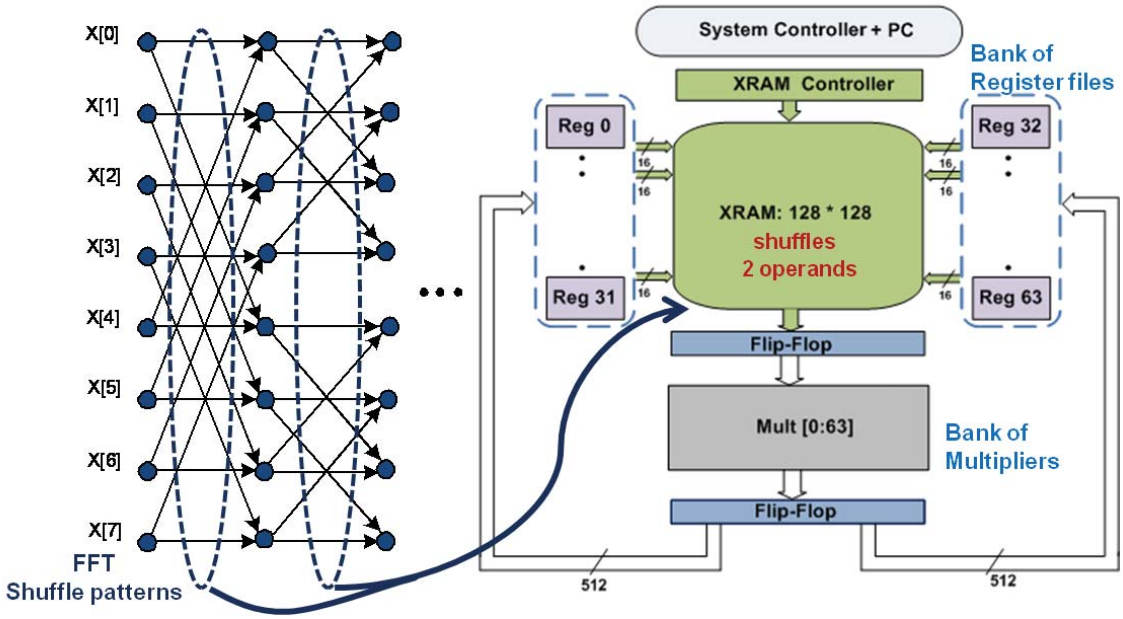

Figure 2.3: Test prototype with XRAM as swizzle network

number of control signals that are required to set the connectivity at the cross points. To mitigate this, XRAM uses a technique similar to what is employed in SRAM. In an SRAM array, the same bit-line is used to read as well as write a bit cell. Until the XRAM is programmed the output buses do not carry any useful data. Hence, these can be used to configure the SRAM cells at the cross points without affecting functionality. Along a channel (output bus), each SRAM cell is connected to a unique bit-line comprising the bus as shown in Fig. 2.2. Cross points along the top row are programmed by the first bit, those along the second row by the second bit and so on. This allows programming multiple cross points in a column (as many bit-lines available in the channel) simultaneously.

Switch fabrics have traditionally been highly interconnect dominated, rendering a significant amount of logic space under-utilized. In a 128×128 fabric with 16 bit channels synthesized using industrial standard libraries in a state of the art 65nm technology, the silicon utilization is found to be only 18%. XRAM mitigates this to

some extent by re-using output channels for programming resulting in improvement of silicon utilization to 45%. To further improve silicon utilization, multiple SRAM cells can be embedded at each cross-point to cache more than one shuffle configuration. In a 65nm implementation, a 16-bit bus width allows six configurations to be stored without incurring any area penalty. Any one of these configurations can be selectively programmed or used to shuffle data. As a result, XRAM can switch configurations in consecutive cycles without any additional delay. Besides, most signal processing applications use a small number of permutations over and over again [52][53][54]. By caching some of those (that are most frequently used), XRAM saves power and latency that would have otherwise been incurred in a conventional network.

2.4 XRAM operation

XRAM operates in 2 modes: *programming* mode and *transmission* mode. In *programming* mode, the controller sends a one-hot signal onto each output bus. A global word-line is then raised high to write to the SRAM cells at cross points and program the XRAM. With 16-bit buses, a 16×16 XRAM can be programmed in a single clock cycle. Larger XRAMs are divided into multiple sections with independent word-lines and one section is programmed at a time. In the test chip, the 128×128 XRAM with 16 bit buses is divided into 8 sections with independent word-lines. To program a channel, in the first cycle all word-lines are raised high while sending an all zero code(16'b0) on the channel. This writes a logic low in all the bit cells in that channel. In the next cycle, a one hot signal is sent while selectively raising only one word-line high to allocate the channel. For instance, to allocate the channel to input 43, the third word-line is raised high while sending 16b0000100000000000 on the channel. Because configurations are never read out, the access transistors in SRAM are upsized to achieve better write robustness.

In *transmission mode*, the word-line stays low and incoming data is passed onto

the output bus at the cross point storing a 1 using a pre-charge followed by conditional discharge technique. By storing multiple 1s in cross points along a row, input data can be multicast to multiple outputs. During the positive phase of clock, input data is launched and the bit-lines are pre-charged to logic high. During the negative phase of clock, the bit-lines are selectively pulled down if the data is high and the cross-point stores a 1. A bank of sense amplifiers evaluates the bit-lines to retrieve data. The bit-lines need not be pulled down all the way to VSS, thereby saving power and improving performance. Because bit-lines either stay pre-charged or get partially discharged, the worst case switching scenario is eliminated greatly reducing capacitive cross talk and enabling lesser spacing between adjacent bit-lines (In the test prototype, input and output wires are routed at 2× minimum spacing). However, this technique results in power dissipation even with a non-switching input that stays high because the bit-lines get pre-charged and discharged every cycle. To mitigate this, incoming data is transition encoded at the input of XRAM. The original data is retrieved back at the output using transition decoders.

2.5 XRAM and conventional fabric

The number of wires and hence the dimension of XRAM grows as O(n), where n is the number of inputs/outputs, compared to O($n(logn)$) in conventional switch fabrics. Simulation results in Fig. 2.4 and Fig. 2.5 compare a conventional fabric that is optimally designed using standard cells and XRAM with increasing dimensions. Xbar dimension is defined as the number of input/output ports. XRAM is 3.3× smaller, 37% faster, and saves 75% energy per bit transfer in comparison with a conventional fabric for a 128 × 128 network. With increasing dimension (both ports and width of buses) XRAM benefits become more apparent, making it suitable for use in systems with high bandwidth and low latency requirements.

XRAM does not require clustering of unique bits from different buses at input,

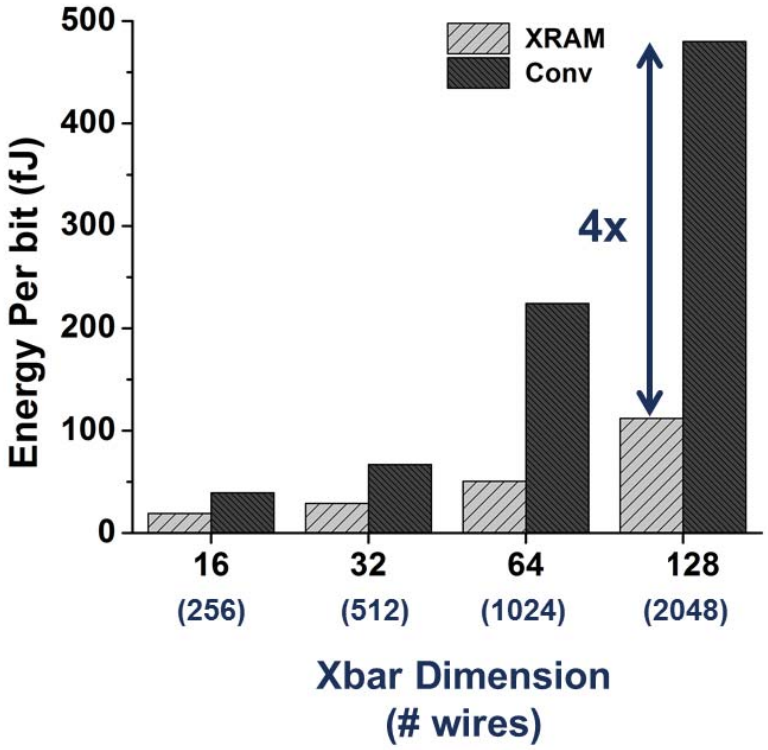

Figure 2.4: Energy comparison of XRAM and conventional fabric with increasing dimension

thereby avoiding routing congestion. Besides, the absence of additional control-overhead results in significant area reduction. This reduces interconnect capacitance and improves performance. Programming power and latency is further reduced by caching frequently used shuffle patterns in XRAM.

2.6 SIMD scalability with XRAM

Switch fabrics are used in all SIMD processors for operand permutation. Most signal processing algorithms are parallel in nature and hence can exploit availability of additional SIMD lanes for higher throughput or energy savings by scaling down supply voltage. Fig. 2.6 and Fig. 2.7 shows the average energy spent per cycle while computing a 1024 point FFT in systems with varying number of SIMD lanes at

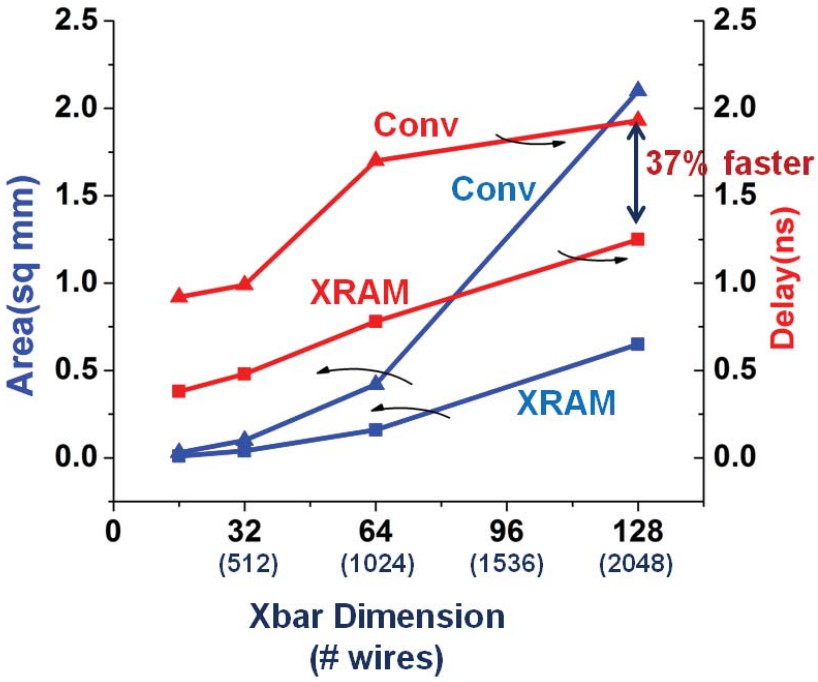

Figure 2.5: Area and delay comparison of XRAM and conventional fabric with increasing dimension

constant throughput. The delay and energy for a conventional mux-based fabric does not scale well and hence does not provide room for voltage scaling with increasing SIMD width. In contrast, an XRAM equipped system saves significant energy with additional lanes because of voltage scaling.

2.7 Circuit implementation

2.7.1 Crosspoint bit cell

Fig. 2.8 shows the cross point circuitry. One of the bits (out0<0>) from the output bus is used to program the cross point. The one-hot signal Config[0:5] is used to select a configuration. The output bus is pre-charged concurrently while incoming data settles on the input bus (during negative phase of Discharge). Thereafter, (during positive phase of Discharge) the output lines are discharged if the cross point bit

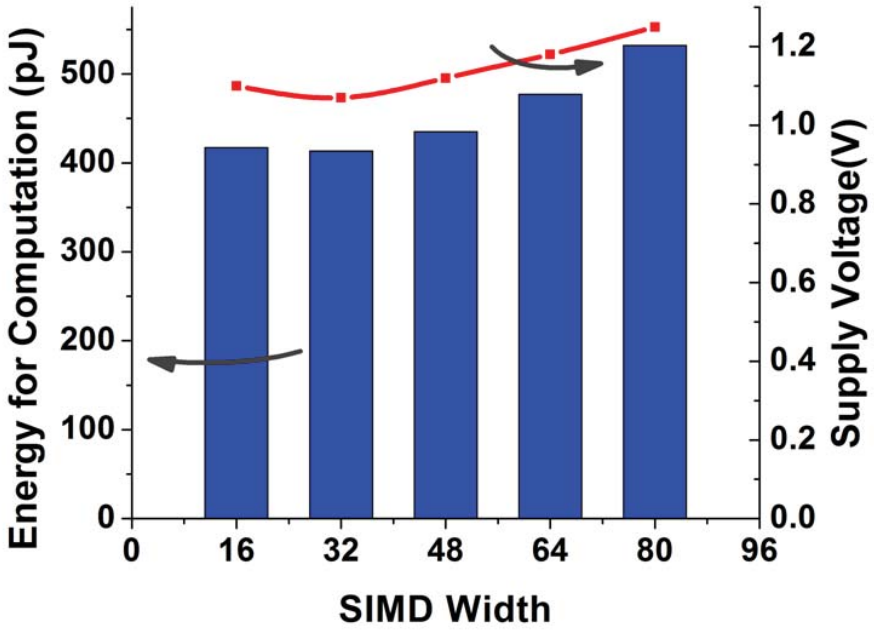

Figure 2.6: Convetional SIMD operating voltage and energy dissipation with increasing number of lanes

cell stores a 1 and the incoming data is 1. Widths of 480nm for the top transistor and 540nm for the bottom transistor were chosen in the discharge stack to optimize overall XRAM delay. In most cases the 2 operands in SIMD data path undergo different permutations. Hence, two sets of configuration select signals are routed in XRAM to provide independent shuffle patterns to either operand. Fig. 2.9 shows the crosspoint layout.

2.7.2 Thyristor based sense amplifier

Output lines are evaluated by a bank of sense amplifiers. Conventional sense amps use a differential amplifier topology for bit-line evaluation. They are very prone to functional failures due to device mismatch. Incorrect evaluation might also result if they are fired too early before an adequate voltage difference has developed on the bit-line. Hence, they rely on accurate timing and vastly oversized devices for

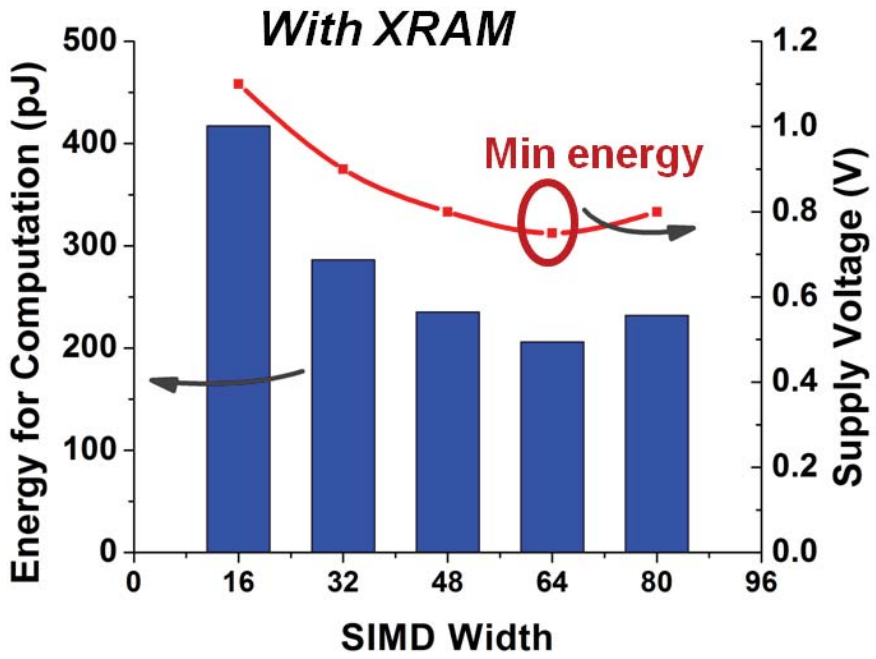

Figure 2.7: XRAM enabledl SIMD operating voltage and energy dissipation with increasing number of lanes

correct operation across different process corners. Besides, they consume power in pre-charging internal nodes every cycle, because during evaluation one of the internal nodes is always discharged irrespective of the bit- line voltage. In SRAM arrays the area and power penalty for such sense amps gets amortized because of column multiplexing. However, in XRAM all output buses are evaluated simultaneously. Because of the large number of sense amps required (2048 for 128 16 bit buses in the test chip), differential amplifier based topology is not suitable. In addition, to enable single ended sensing a reference voltage would be required. To mitigate these shortcomings, a novel single-ended thyristor-based topology is used to design sense amp as shown in Fig. 2.10. The sense amp is initially pre-charged to a low gain leakage state as shown in Fig. 2.11. In this stage the thyristor node voltages are weakly held because of leaking pre-charge devices. During evaluation, internal node A is coupled to the output line through a PFET access transistor. If the output line

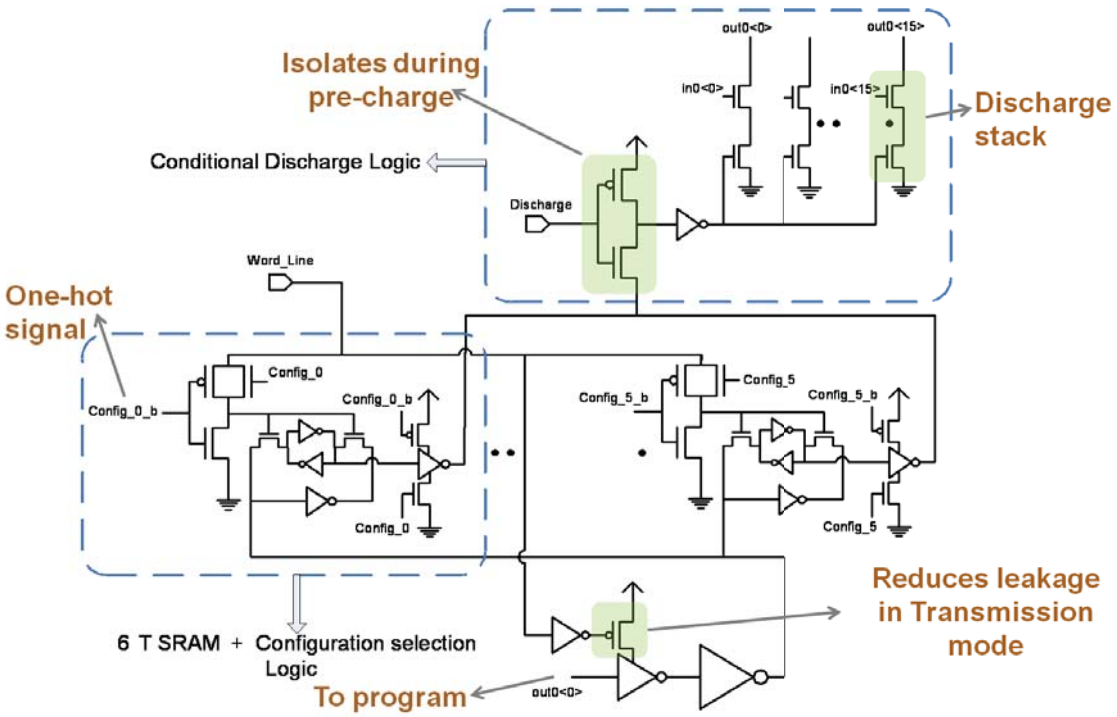

Figure 2.8: Crosspoint schematic

stays pre-charged, the thyristor continues to stay in its low gain leakage state. The pre-charge transistors are sized large enough to compensate for leakage through the thyristor pair.

On the other hand, if the output line is discharged by a cross point, node A initially follows the output line voltage. The PFET in the thyristor gradually turns on which subsequently turns on the NFET producing a regeneration effect that causes the thyristor to switch into its high gain stable state as shown in Fig. 2.12. The PFET access transistor is sufficiently weak to prevent the output line from fully discharging to save power. Because input evaluation does not involve any contention, this topology is more tolerant towards mismatch induced functional failure.

Input evaluation in the thyristor based topology is slower than a conventional sense amp. This is amortized by firing the sense amps as soon as bit-lines start discharging as shown in the timing diagram in Fig. 2.10. This simplifies timing gen-

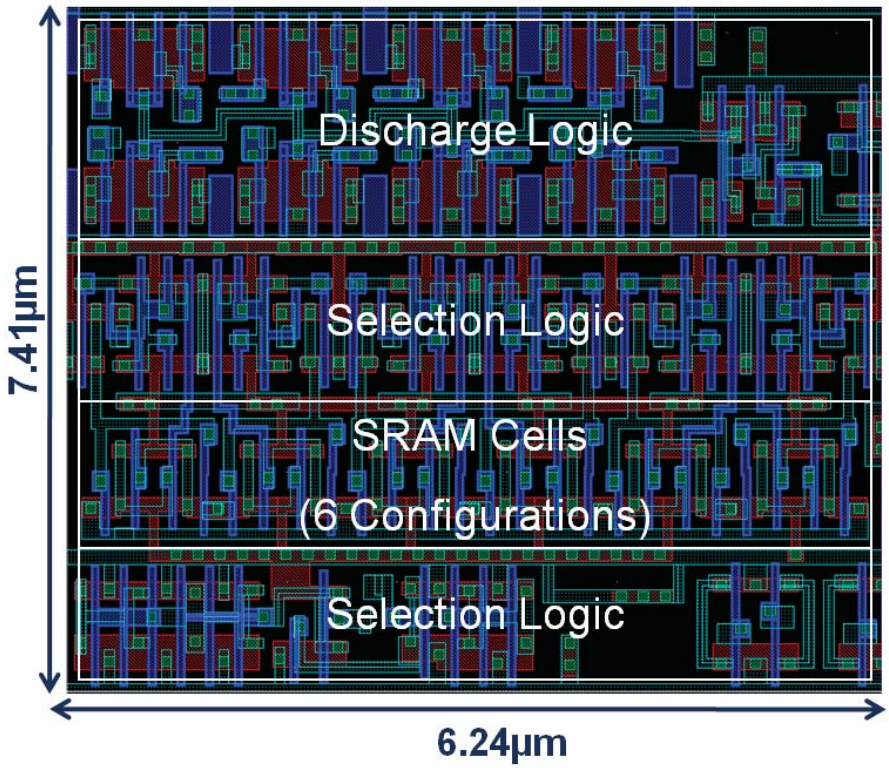

Figure 2.9: Crosspoint layout

eration since SE and Discharge can be generated off the same edge. Given improved mismatch tolerance, devices can be downsized, resulting in only 3% to 20% increase in XRAM area in comparison with 12% to 52% for a conventional sense amp across dimensions ranging from 128×128 to 16×16. The internal nodes switch only if the input discharges, thus reducing pre-charge power. As can be observed from the cross point and sense amp circuits, only falling edge transitions are critical. Hence delay is further improved by skewing devices. Fig. 2.13 shows the sense amplifier layout.

2.8 Test Prototype

The test prototype is fabricated in TSMC 65nm, 9 metal stack bulk CMOS process using a semi custom design flow. The cross point and sense amp circuits are custom laid out. As shown in Fig. 2.8, the circuit is similar for all 16,384 cross points except for

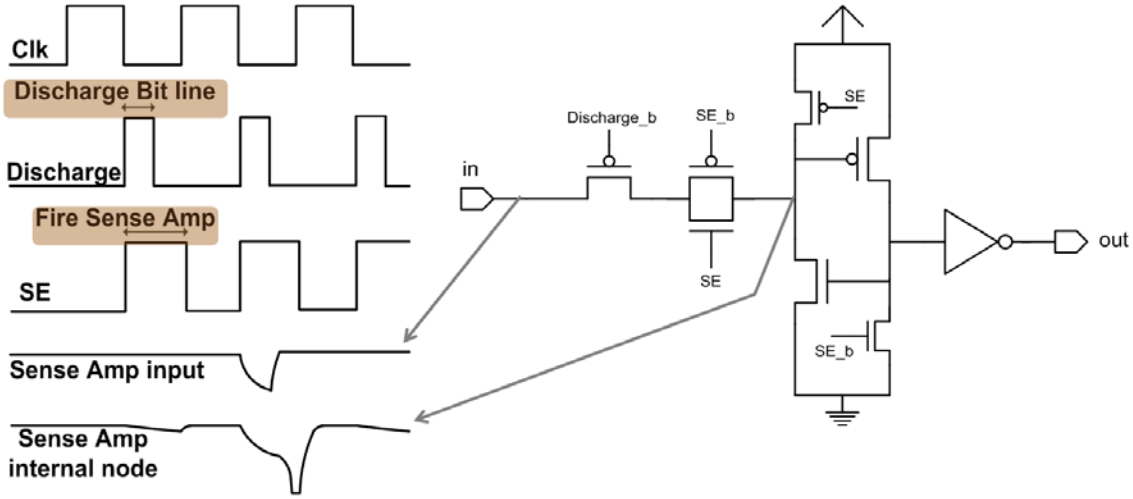

Figure 2.10: Thyristor based sense amplifier with timing diagram

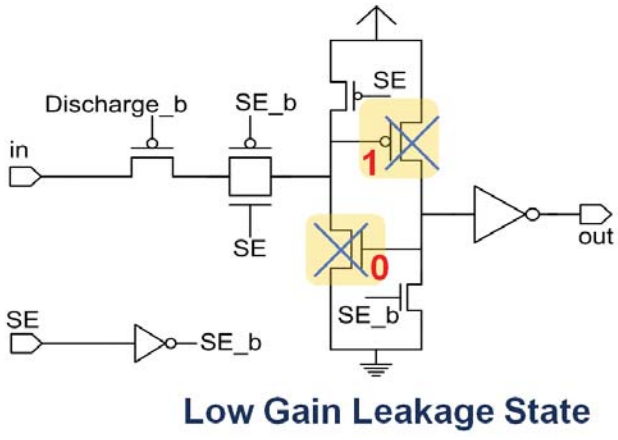

Figure 2.11: Thyristor based sense amplifier: Low gain state

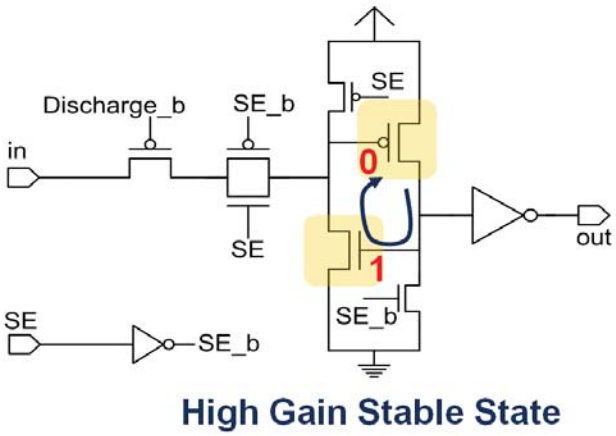

Figure 2.12: Thyristor based sense amplifier: High gain state

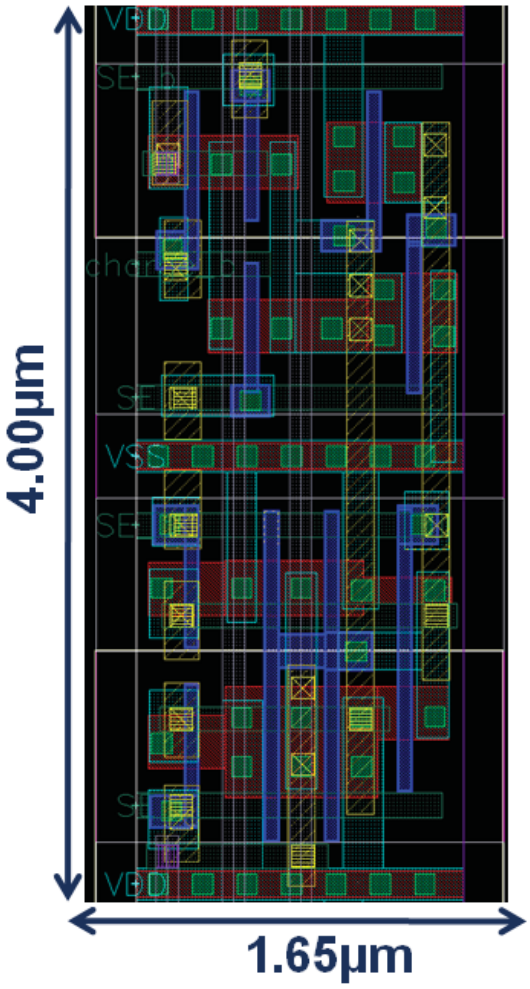

Figure 2.13: Thyristor based sense amplifier layout

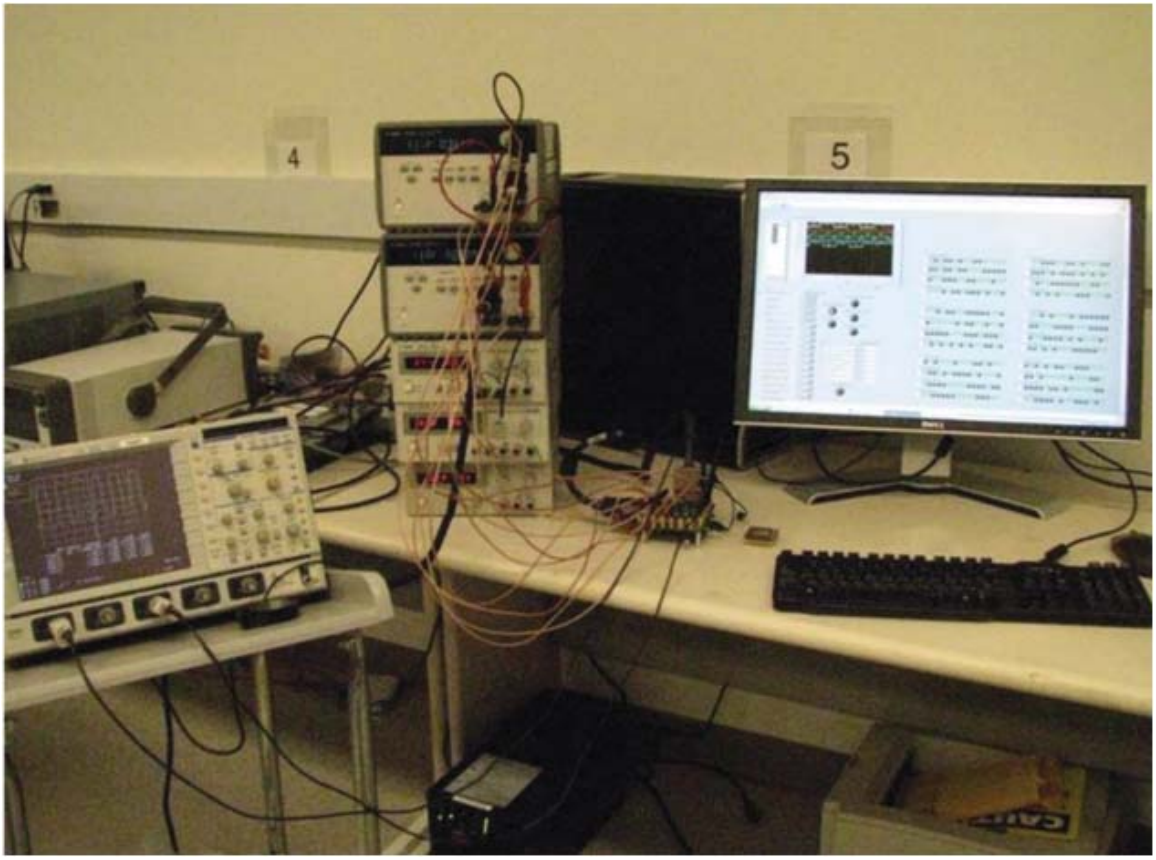

Figure 2.14: Measurement set up

the output line that is used for programming them. A skill script is used that modifies a generic cross point layout to create all cross points. The cross points are then tiled together along with input/output drivers as is done in SRAM arrays. The input buses for XRAM are laid in metal layers 2 and 4, while the output buses are laid in layers 3 and 5 at 2x minimum pitch. Clock, Discharge and SE are distributed through H-trees in layer 6 to minimize capacitive coupling from signal routes. Transistor sizing is based on critical path delay (delay for data transfer from top-most to right-most cross point) optimization at the typical corner and functional verification at all process corners. Other modules in the chip are synthesized using Synopsys Design Compiler and auto placed and routed with Cadence SoC Encounter. Timing is met at 2.5ns with a global clock skew of 180ps.

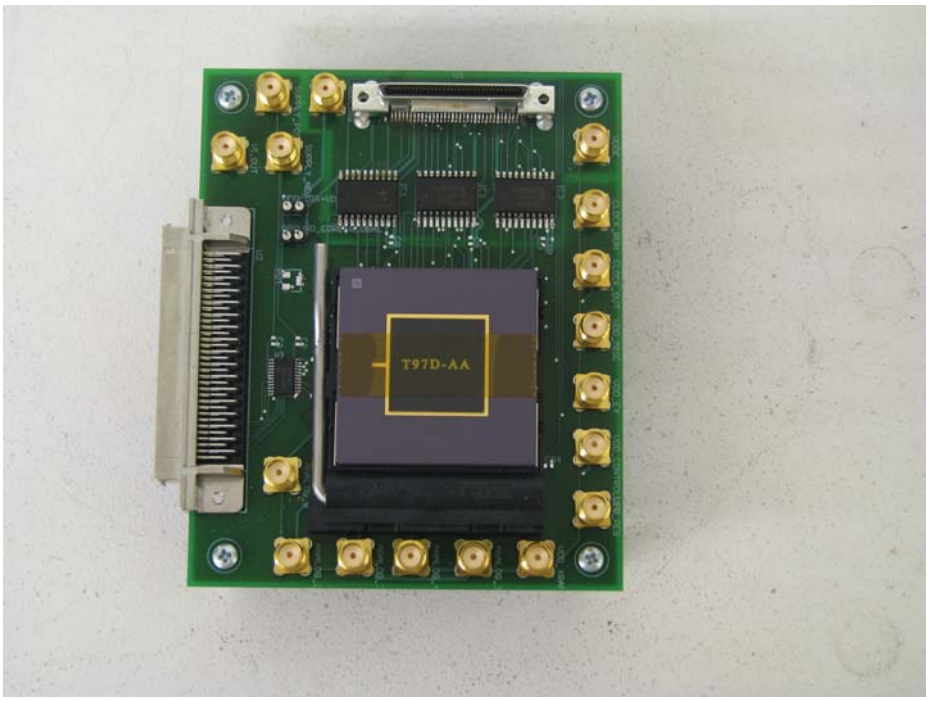

Figure 2.15: PCB hosting XRAM enabled SIMD prototype

Fig. 2.14 shows the test setup. For fairness in comparison, two prototypes are tested: a 64 lane SIMD with a 128×128 XRAM and a 16 lane SIMD with a 32×32 conventional crossbar. Both chips have five core(1.1V) and one IO(2.5V) voltage domains. The switch fabric, execution stage, register files have separate voltage domains for independent power measurement. Other miscellaneous peripherals and scan are assigned the fourth core voltage domain. The clock generator is placed on the fifth power domain to avoid power supply noise because of processor activities affect operation frequency. Each power grid is designed for 50mV voltage drop under worst case scenario. For XRAM, most number of adjacent cross points fire during a broadcast causing maximum supply droop. 66 of the 113 pins in the 64 lane SIMD chip and 36 of the 57 pins in the 16 lane SIMD chip are dedicated for supply. Fig. 2.15 and Fig. 2.16 show the printed circuit boards with both test prototypes.

Die micrographs of the 16 Lane and 64 Lane SIMD processors are shown in Fig. 2.17 and Fig. 2.18 respectively. The 16 lane system has a 32×32 synthesized

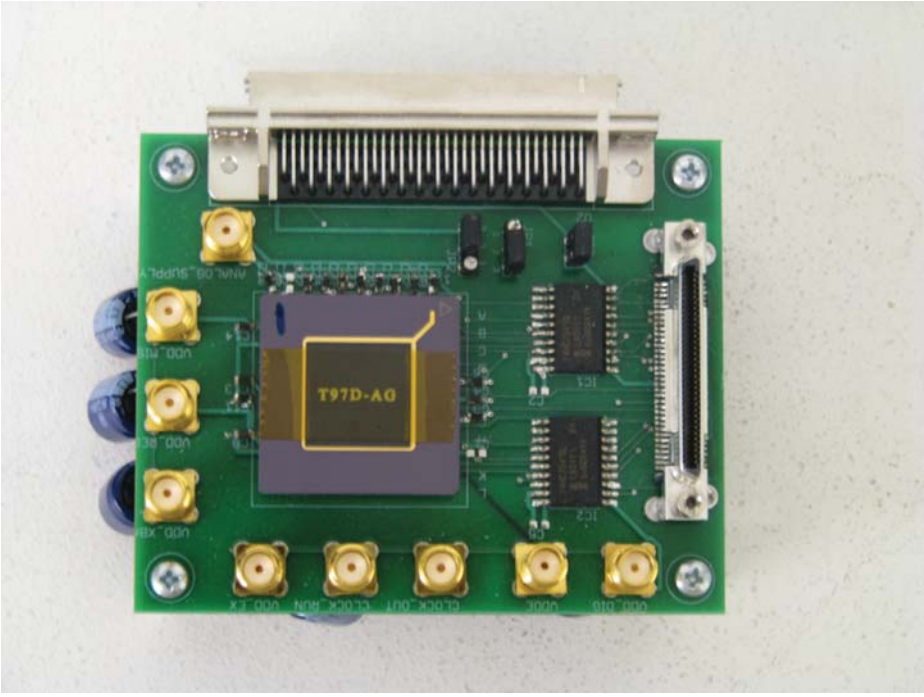

Figure 2.16: PCB hosting conventional SIMD prototype

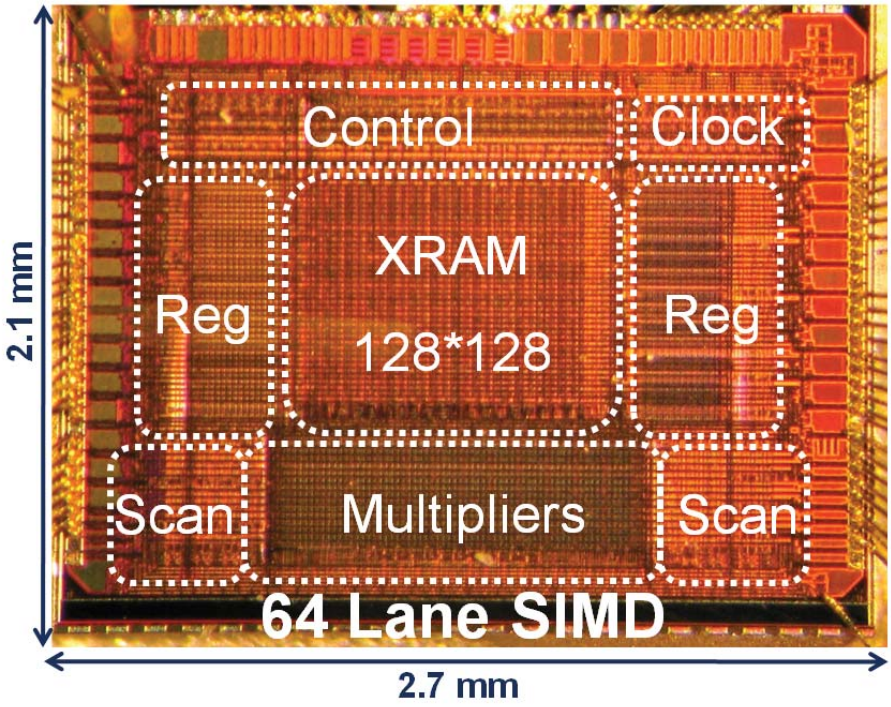

Figure 2.17: XRAM enabled SIMD die photograph

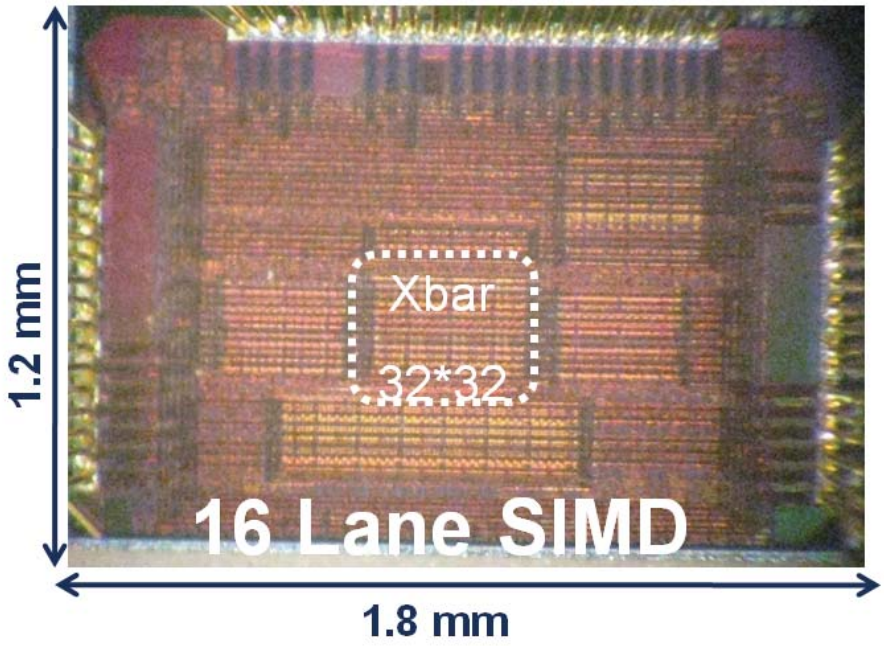

Figure 2.18: Conventional SIMD die photograph

switch fabric that spans 0.16mm^2, which is 7.5% of the total die area (2.16mm^2). The 128×128 XRAM in the 64 lane system spans 0.85mm^2, which is 14.9% of chip area (5.7mm^2). A synthesized fabric of similar dimension would span 2.2mm^2 and would require 3$\times$ more energy per bit transfer. Besides, XRAM topology allows source modules to be placed on either the left or right side and destination modules on either the top or bottom side of the fabric, making chip floor planning flexible. This allows splitting the register files on either side of XRAM as shown in the 64 lane SIMD die photo.

Both chips use a scan based scheme for transferring data in and out of the processor. The scalar pipeline that acts as the host processor for programming the XRAM can be run in test mode through scan. A 128 entry FIFO that is used as instruction cache and the 16 entry register file are directly programmable through the scan chain. In configuration mode the scalar processor is clocked from an external source for programming the crossbar and predication registers used by SIMD instructions. The processor is clocked (externally or internally from clock generator) for a given

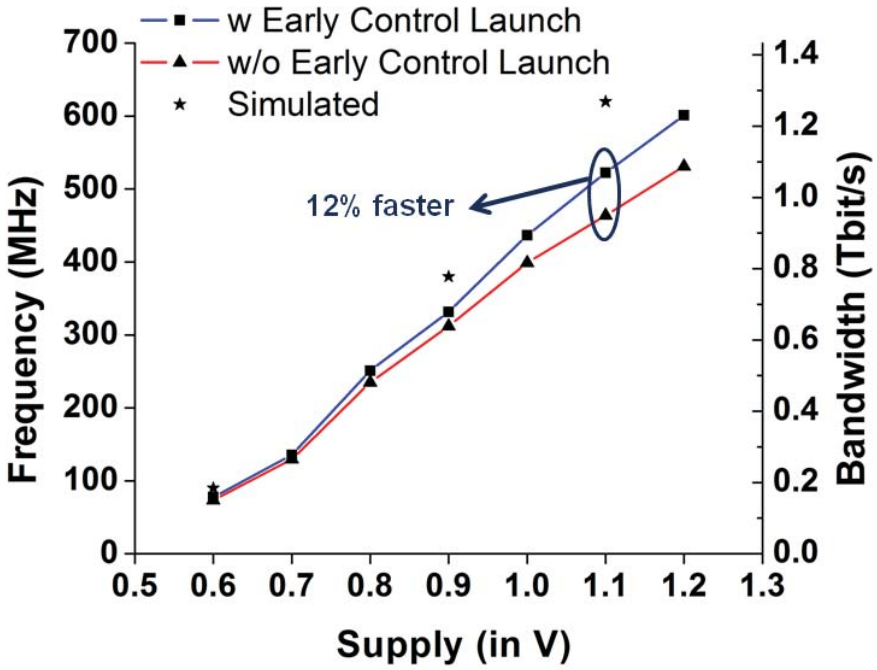

Figure 2.19: Measured XRAM speed and throughput

number of cycles and data can be read out from the register files to verify for correctness. The execution stage with 16 bit multipliers is the critical path that sets the processor frequency. For independently testing the switch fabric at higher speed, the register files and execution units are augmented with LFSRs and signature analyzers. The LFSRs and signature analyzers are designed for operating as fast as 700MHz since the maximum target frequency for XRAM is 650MHz. The pipeline registers between the register file and switch fabric are also made scanable to verify that correct data gets launched into the switch fabric. The Clock, Discharge and SE signals are generated on chip with programmable delays controlled by the scan chain and off-chip analog biases to modulate the position and duty-cycle for Discharge and SE. These high frequency signals are made observable through an on chip voltage to current (VI) converter that can source a large current so that a voltage pulse is made available as a current pulse external to the chip. The current pulse when passed through a resistor of appropriate value recreates the voltage waveform.

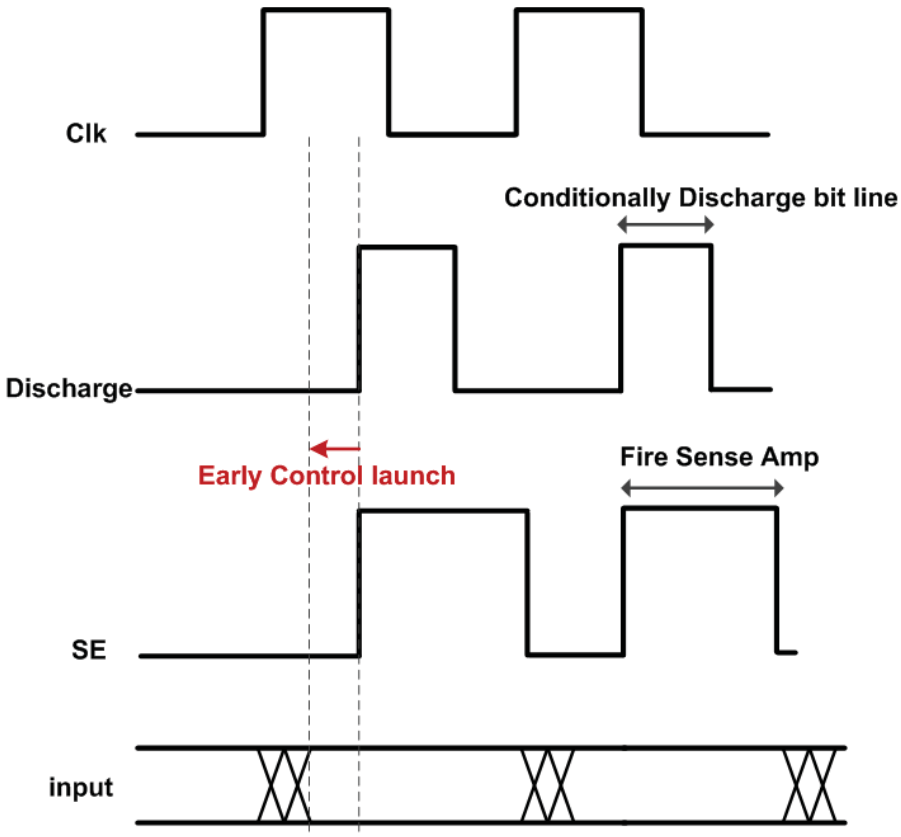

Figure 2.20: Early control launch timing diagram

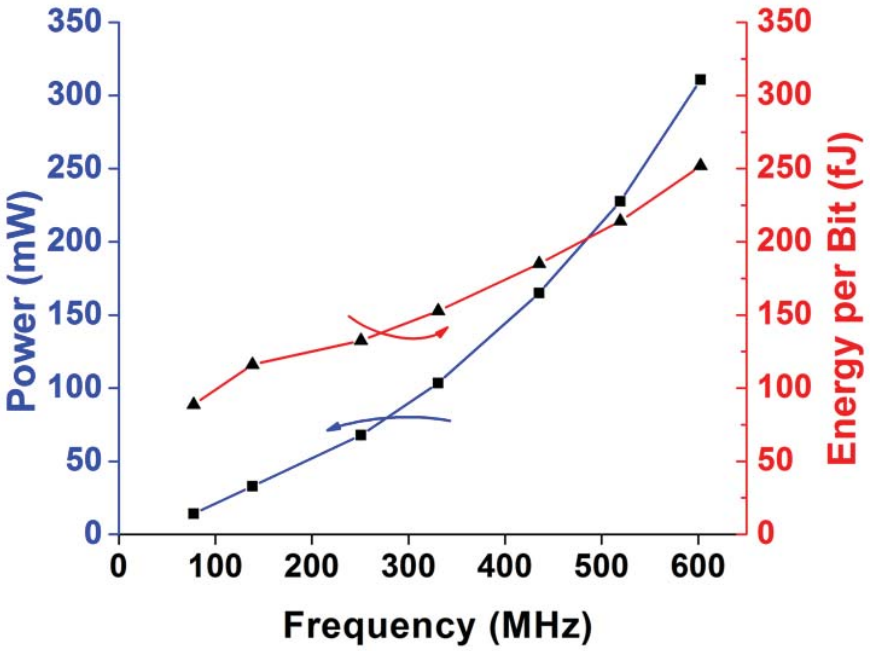

Figure 2.21: Measured XRAM power and energy efficiency

2.9 Measurement Results

The test chips are verified functionally by running a 64 point FFT that has 6 butterfly stages. The shuffle pattern for each butterfly stage is configured into the switch fabric so that the processor can consistently process data without halting. The switch fabrics are tested by streaming data in various permutations and verifying their signatures. Measurement results for the 128 × 128 XRAM for supply ranging from 600mV to 1.2V are shown in Fig. 2.19 and Fig. 2.21. XRAM functionality is limited by sense amps that do not operate reliably below 580mV. XRAM has a peak bisection bandwidth of 1.07Tb/s at 1.1V while consuming 227mW at 20% switching activity. This translates to an efficiency of 4.71Tb/s/W and 212fJ per bit transfer.

The delay through XRAM can be analyzed in two phases. In the first phase (defined by Discharge staying low), input data settles and various control signals (that select configuration, mode of operation etc.) propagate to the cross points. During

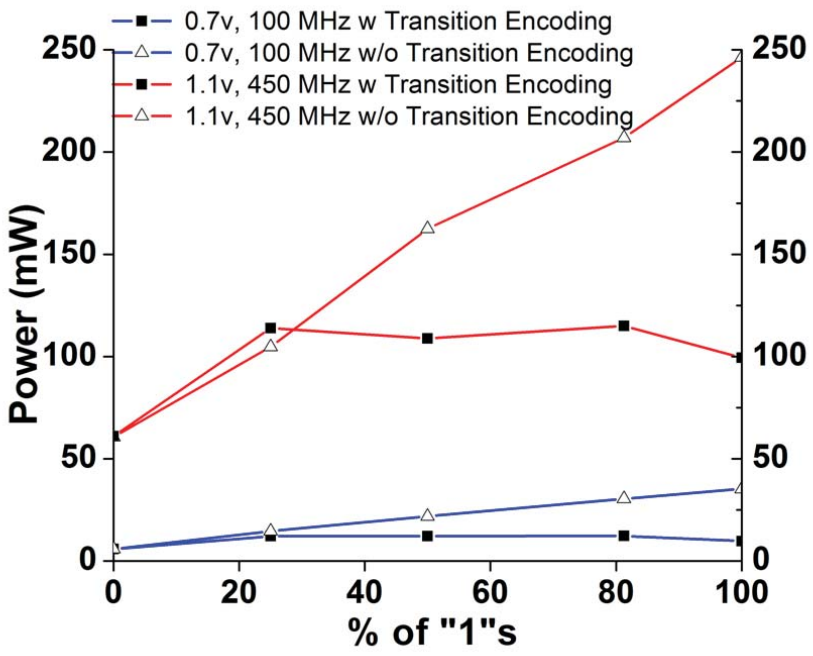

Figure 2.22: Measured XRAM power with and without transition encoding

the second phase (defined by Discharge staying high), output lines are selectively discharged and evaluated in the sense amp. At nominal supply (1.1V), the delays for both phases are balanced. In this case a 12% speed improvement is achieved when the first phase delay is optimized by launching XRAM control signals early by exploiting useful skew as shown in Fig. 2.20. At lower supplies, output line discharge followed by sense amp evaluation slows down significantly making the second phase delay dominant. Hence optimizing the first phase delay results in no significant overall speed improvement.

Measurement result showing the impact of transition encoding is shown in Fig. 2.22. With transition encoding disabled XRAM power increases linearly with increasing percentage of 1's in the input data. In contrast,with transition encoding enabled energy savings of 33% and 44% are observed at 1.1V and 0.7V respectively. The minima at either extreme are due to zero switching activity in the input/output buses. Transition encoding saves energy by eliminating discharge of output buses when in-

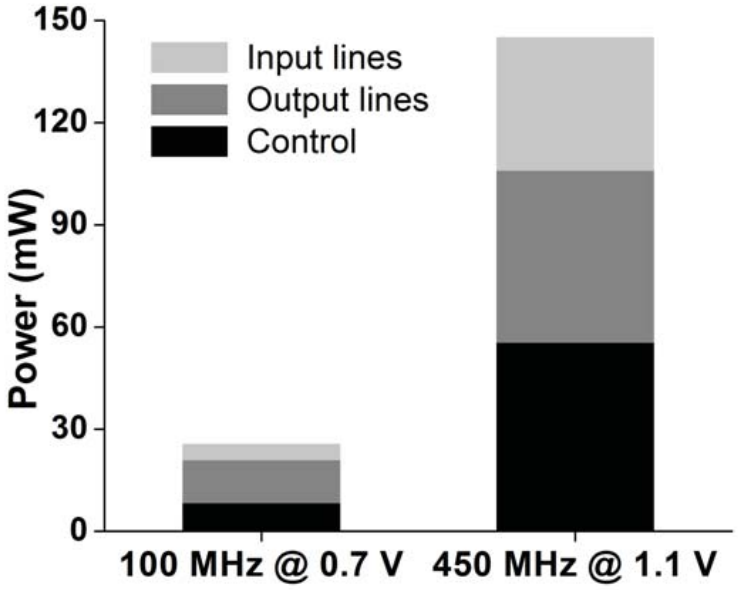

Figure 2.23: Measured XRAM power breakdown

put is statically held high. The power breakdown in Fig. 2.23 reveals that output lines dominate XRAM power at low voltage. This can be accounted for due to the fact that, energy dissipation in output lines scale differently from other modules with voltage scaling. The threshold voltage of the PFET in thyristor at the typical corner is 330mV. So the thyristor triggers into the high gain state when the output line discharges by at least 330mV irrespective of the supply voltage. By reducing supply from 1.1V to 0.7V, expected energy savings from other circuits (clock buffers, cross points, input/output drivers etc. where energy dissipation is C.Vdd^2) is 2.5× (1.12/0.72). For output lines the energy dissipation is C.Vdd.ΔV. So, expected energy saving is only 1.6×(1.1/0.7). However, only the output line that is sensed by the latest triggering sense amp discharges the least. The duration for Discharge and SE remains the same for all cross points and sense amps. So other output lines continue to discharge even after the sense amps sampling those trigger. Effectively, expected energy savings on output lines should be between 1.6× and 2.5× depending on intra-die process

METRIC		16 Lane SIMD	64 Lane SIMD	Gains
Power (mW)	Ex	50.60	20.16	2.51x
	Reg. File	40.70	18.72	2.17x
	Xbar	47.30	33.84	1.40x
	Control	13.20	8.64	1.53x
	Total	151.80	81.36	1.87x
Frequency(MHz)		400	100	-
Supply (V)		1.10	0.72	-
Throughput (GOPs/Sec)		6.4	6.4	-
Efficiency(GOPs/W)		42.4	78.7	1.87x

Figure 2.24: Measured power and performnace for 16 and 64 lane SIMD

variations. With less variability, energy savings would be close to 1.6× and conversely with more variability the ratio would move towards 2.5×. This explains the power trend shown in Fig. 2.22 and justifies transition encoding being more effective with supply voltage scaling.

Measured results for XRAM equipped 64 lane and conventional 16 lane SIMD processors are summarized in Fig. 2.24 under two extreme work load conditions. The 16 lane system runs at 400MHz at 1.1V. For similar throughput the 64 lane system is voltage scaled down to 720mV to operate at 100MHz. In the 64 lane system, energy savings in register files and execution units because of low voltage operation supersede the additional energy required to transfer data over a bigger fabric resulting in better system efficiency. For FFT like light load applications, the 64 lane system consumes 40.32mW which is 29.7% less energy than the 16 lane system. For matrix inversion like heavy load applications power consumption is 81.36mW and 46.8% energy is saved (at iso-throughput run time for both processors is same, hence %power saving is equivalent to %energy saving).

Comparison of XRAM with a prior art [48] that uses flash programmable pass

Metric	XRAM	[48]
I/O Ports	128+128	8+8
Total Bits	2048	512
Configs	6	8
Storage	SRAM	FPT*
Tech.	65nm	180nm
Area(mm^2)	0.87	1.38
Freq.(MHz)	523	100
Latency(ns)	1.89	8.0/22.0**
Supply(v)	1.1	1.6/2.0***

*Flash Programmable Pass Transistor
**worst case
***Programming voltage

Figure 2.25: Comparison of XRAM with a recent switch fabric

transistors to store permutation pattern at cross point switches is shown in Fig. 2.25. Although [48] has 8 configurations, it takes longer to program the fabric than XRAM (write duration for flash cell is a lot higher than SRAM). It requires multiple cycles for data to traverse across the fabric. Besides, [48] does not use bulk CMOS process and hence cannot be easily integrated in CMOS designs. Generation and handling of a higher programming voltage for configuring flash cells also require significant energy and logic overhead. XRAM fares better in performance (4.2× faster), area as well as energy efficiency. Fig. 2.26 and Fig. 2.27 show the delay distribution for all XRAM lanes at nominal and reduced supply voltages respectively.

2.10 Summary

In this chapter, we presented a fast, low power, area efficient circuit technique called XRAM to design high radix interconnect fabrics for multi-processor systems.

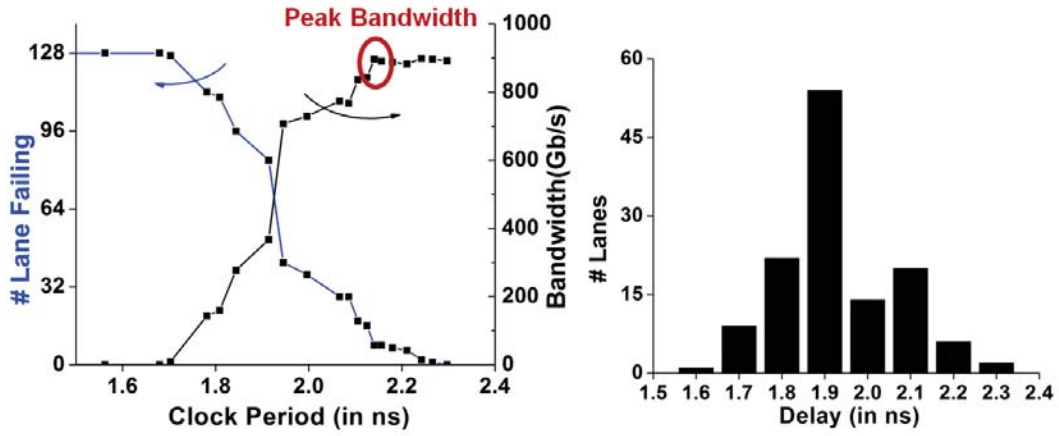

Figure 2.26: XRAM variability measurement at 1.1V

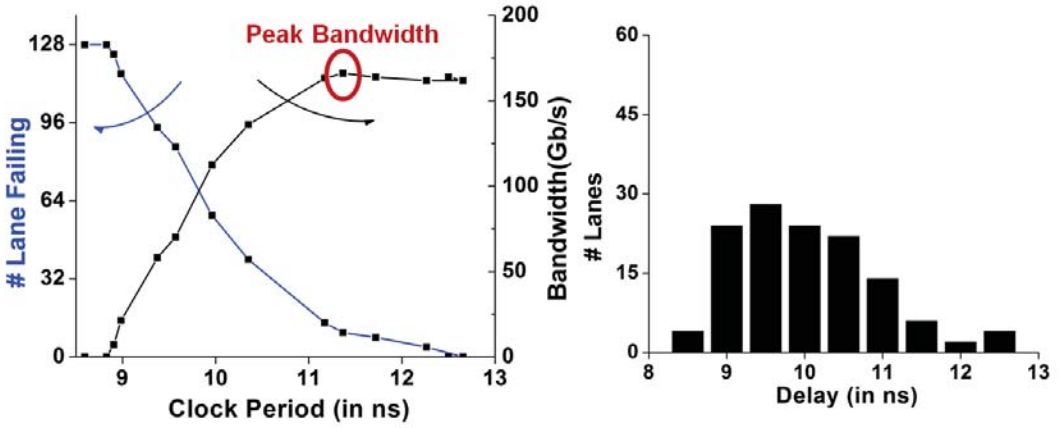

Figure 2.27: XRAM variability measurement at 0.6V

In contrast with prior switch fabrics that are made from elementary 5×5 switches, we demonstrate in silicon a single stage 128×128 fabric that spans $0.85\text{m}m^2$ in 65nm, provides a peak bisection bandwidth of 1.07Tbit/s while operating at 530MHz at 1.1V and dissipates 227mW at 20% switching activity. XRAMs energy efficiency is measured to be 4.72Tb/s/W with 212fJ energy dissipation per bit transfer. Simulation results demonstrate that XRAM equipped SIMD systems can scale to much wider lanes in comparison with conventional SIMD. Silicon results from test prototypes of XRAM aided and conventional SIMD processors fabricated in 65nm demonstrate 47% energy savings for compute intensive workloads.

CHAPTER III

SWIFT : Swizzle Interconnect Fabric Topology

3.1 Introduction

The arbiter is another integral module of a generic interconnect fabric. The swizzle network called XRAM (discussed in chapter 2) uses circuit techniques to optimize routing of data across the fabric. However, it requires a centralized arbiter to program the switch. At higher radices, programming latency limits the operating frequency and hence XRAM's throughput. Besides, communication to and from the arbiter costs additional delay and energy overhead, that limits XRAM's efficiency. To address this, in this chapter we introduce circuit techniques to integrate the arbiter within the permutation network as shown in Fig. 3.1. This fabric (called SWizzle Interconnect Fabric Topology) retains all XRAM data permutation capabilities in addition to its ability to perform high radix arbitration without incurring additional delay. SWIFT is a 32×32 (with 64b data bus) self-arbitrating switch fabric. It achieves a bandwidth of 2.1Tb/s with single cycle arbitration and data transfer latency in 65nm bulk CMOS technology while operating at 1026MHz at 1.2V. SWIFT co-optimizes arbiter and crossbar logic using a unique fabric architecture that integrates conflict resolution with data routing to optimally use logic and interconnect resources. It spans 0.35mm^2, achieves an efficiency of 7.39Tbps/W, and operates down to 530mV.

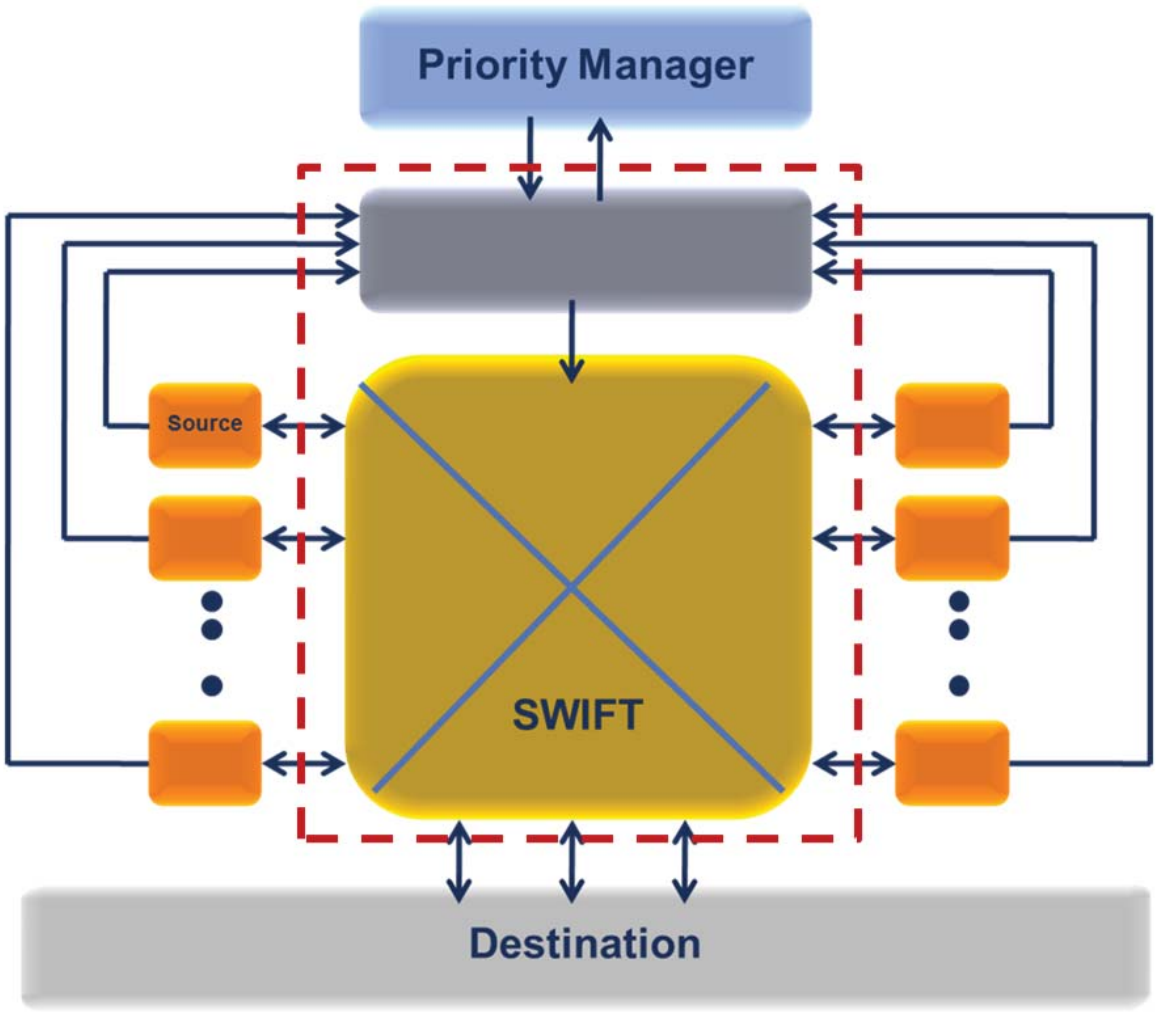

Figure 3.1: SWIFT: Self arbitrating fabric

3.2 SWIFT architecture

High radix, high bandwidth, and low latency switch fabrics are key enablers for high-end servers, Ethernet routers, multiprocessor NoCs, multimedia accelerators, and manycore computers in general [56] [57]. Conventional switch fabrics typically consist of a crossbar to route data and a separate arbiter to configure the crossbar. This poses two hurdles for scalability: 1) Routing to and from the arbiter incurs significant overhead as the number of sources and destinations grows. Also, the rapidly growing complexity of centralized arbitration with increasing crossbar radix can dominate overall delay. 2) Area utilization is poor since the arbiter is logic dominated, whereas the crossbar is routing intensive. The high cost of large radix switches limits fabric scalability [58] [59] by requiring more stages in the data traversal path. This results in higher latency, reduced energy efficiency due to intermediate data storage, and complex routing protocols to handle inter-stage communication.

To mitigate this, we introduce an interconnect fabric called SWIFT(SWizzle Interconnect Fabric Topology). Fig. 3.2 shows the top level diagram of SWIFT fabric. It features a novel distributed arbitration scheme that reuses the data transfer bit-lines as priority lines for conflict resolution and locally stores the connectivity status at crosspoints. This eliminates additional routing and logic overhead to produce a compact design. A 32×32 router with 64b data buses (2048 wires) requires just $0.35\text{m}m^2$ in 65nm, including single cycle arbitration. This corresponds to the area required to route its 2048 input/output wires at $2\times$ minimum spacing (and no additional tracks). SWIFT achieves a bandwidth of 2.1Tb/s at 1.2V with an efficiency of 7.39Tbps/W, and is fully functional down to 530mV with a peak efficiency of 36.8Tbps/W.

SWIFT reuses concepts from SRAM design to make single cycle arbitration and data transfer latency possible at high radices. It uses an area efficient thyristor-based sense amplifier enabled latch (SAEL) for fast robust single-ended bit-line evaluation. It supports four priorities for fairness during conflict resolution and the ability to

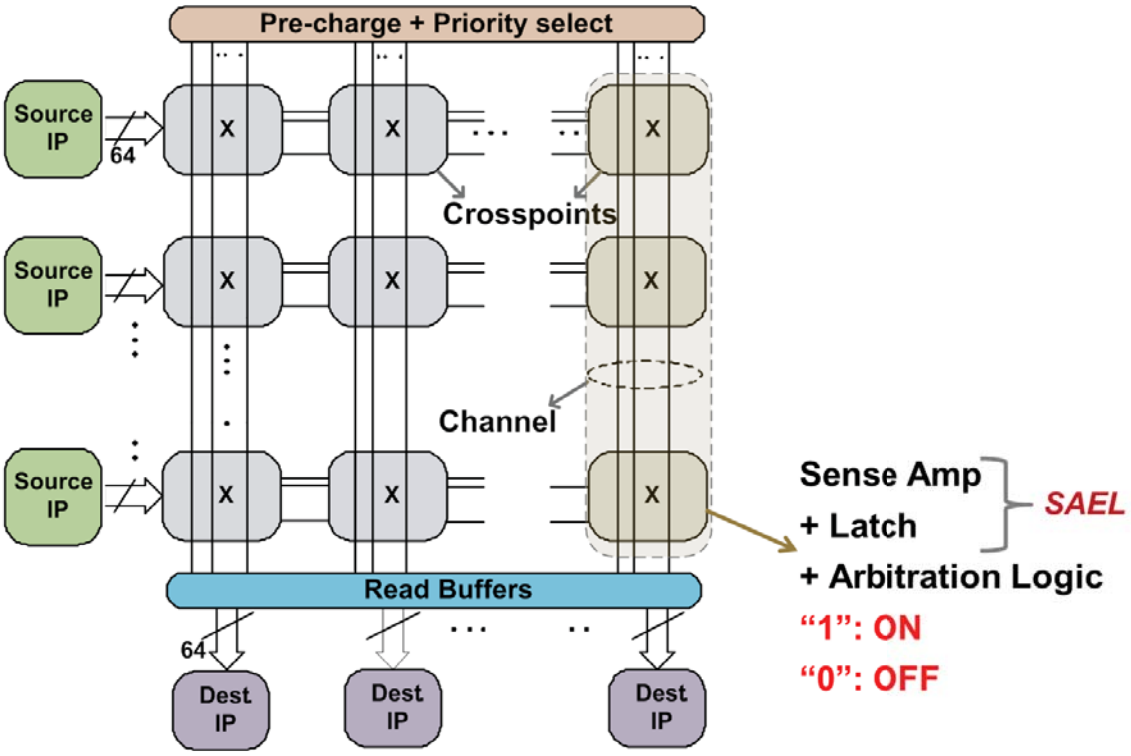

Figure 3.2: SWIFT fabric topology

multicast. Hence, SWIFT is 1.9× more energy efficient with 53% more bisection bandwidth at 80% lower latency over contemporary single stage fabrics.

In SWIFT the input (source IP) and output (destination IP) buses run perpendicularly as shown in Fig. 3.2. This creates a matrix of crosspoints and a connection between an input and output bus is established by locally storing a logic 1 at their crosspoint. At most one logic 1 can exist along a column, whereas multiple 1s can be present along a row to multicast data

3.3 SWIFT operation

Every output channel in SWIFT can independently operate in one of two modes: *Arbitration* or *data-transmission*. A pre-charge followed by conditional discharge scheme is used for high speed signaling on the channels, eliminating worst-case switch-

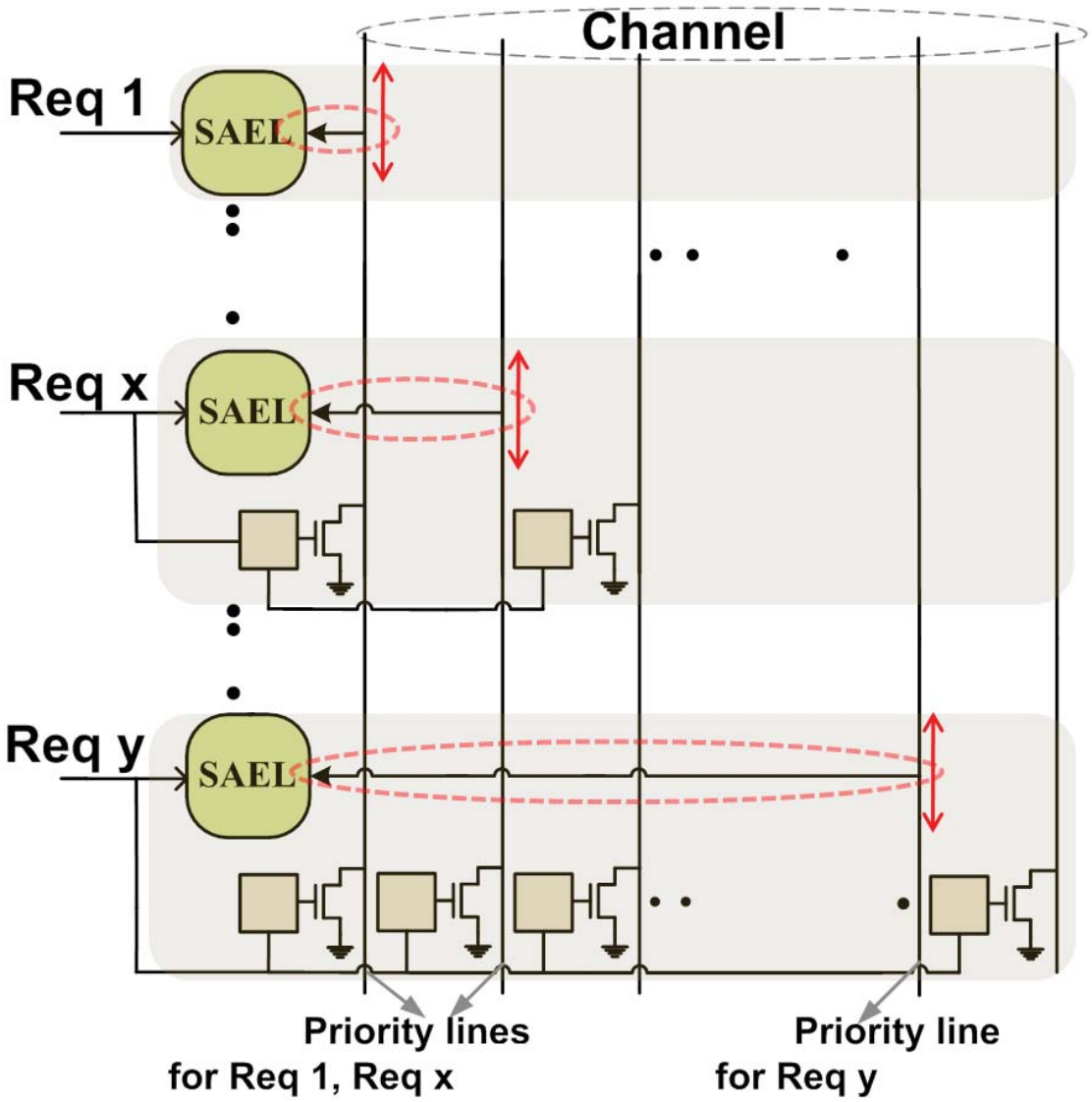

Figure 3.3: Arbitration technique using bit-lines

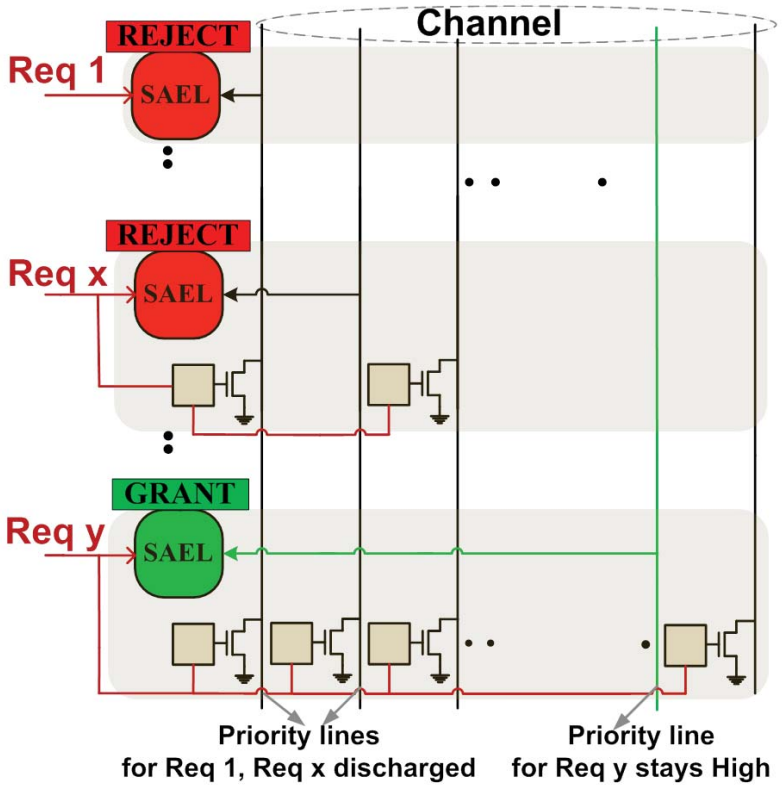

Figure 3.4: Arbitration technique using bit-lines

ing induced crosstalk. In *arbitration* mode, each input channel is assigned a unique bit-line from the output bus as its *priority line*. As shown in Fig. 3.3. the left-most bit-line is the priority line for the top most crosspoint corresponding to *Req 1*. The second bit-line is the priority line for the crosspoint corresponding to *Req x*. Similarly, the second right most bitline acts as the priority line for the crosspoint corresponding to *Req y*. In the first phase of the arbitration cycle all priority lines are precharged high. Upon requesting an output channel, a particular input channel suppresses lower priority channels competing for the same output bus by discharging their priority lines in the second phase of the arbitration cycle. It concurrently samples its own priority line with an SAEL which, if high, guarantees that no higher priority inputs requested the channel. With this technique we can arbitrate among as many requests simultaneously as the number of bit-lines available in the channel.

In this scenario, the priorities are assigned in an increasing order such that *Req*

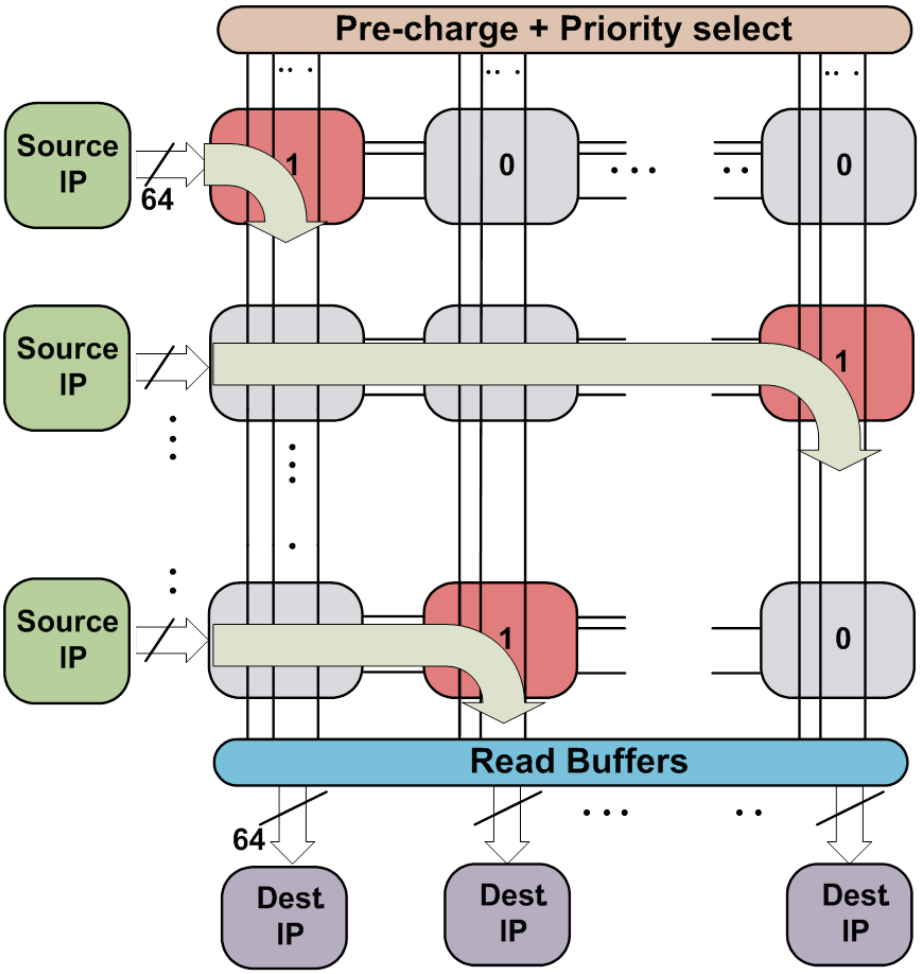

Figure 3.5: Data routing in SWIFT

1 < *Req x* < *Req y*. Hence when *Req 1*, *Req x*, *Req y* are asserted, *Req* y kills *Req x*, and *Req 1* by discharging their priority lines. In this case, a 1 is stored directly at the crosspoint of *Req* y, indicating granting of the channel as shown in Fig. 3.4. Conflict detection and resolution is thus performed in one cycle with the same bit-lines and pull down devices that are meant for data transfer. This obviates the need for additional interconnect and results in a very compact and fast implementation.

For *data-transmission*, bit-lines comprising a channel are discharged at a cross-point if the SAEL stores a logic 1 and input data is high. This is shown in Fig. 3.5. As can be seen clearly, the fabric can seamlessly multicast/broadcast data to multiple destinations by storing multiple 1s in the crosspoints corresponding to those

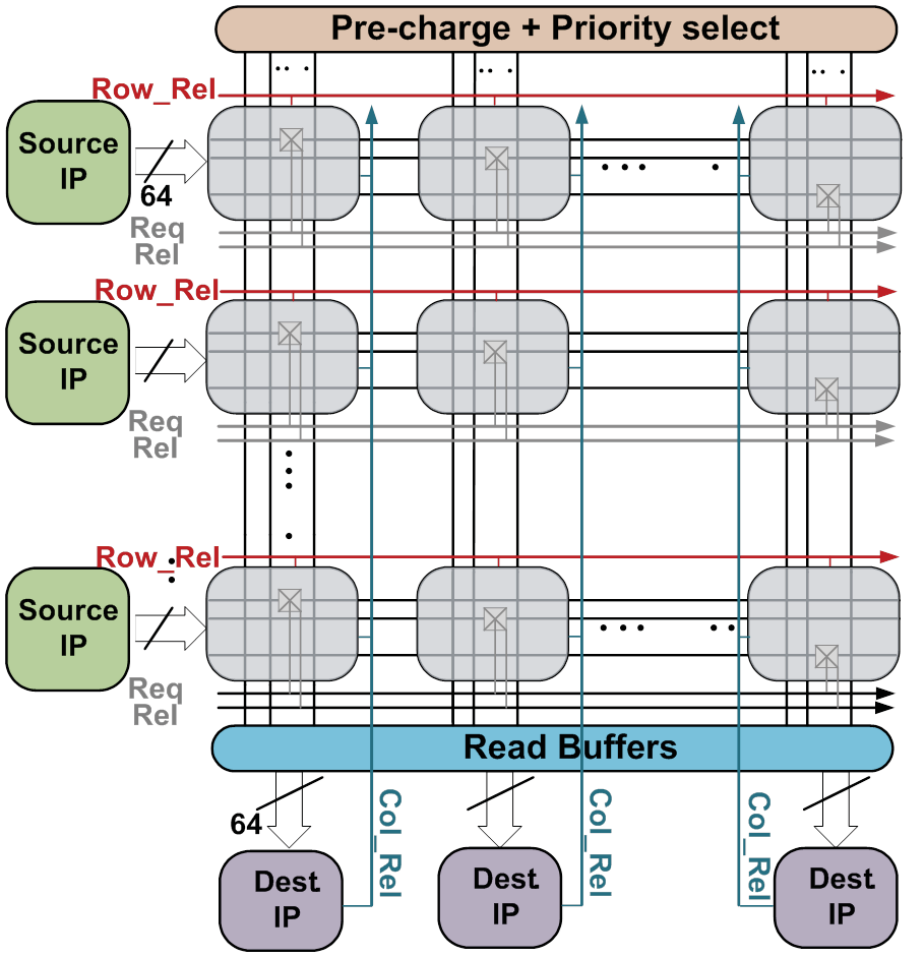

Figure 3.6: Additional SWIFT arbitration control signals

destinations. However, with a static input that stays high, the bit-lines get repeatedly pre-charged and discharged at the crosspoint storing a high. To save power in such a situation, incoming data is transition encoded at the input ports of SWIFT. The original information is retrieved back at SWIFT output ports using transition decoding.

Any channel can independently operate either in the arbitration or data transfer mode of operation. To further optimize interconnect utilization within the fabric a unique protocol is used to request and release channels. Two additional word-lines called *Req* and *Rel* are laid for every input as shown in Fig. 3.6. A source requests a channel by raising *Req* word-line high and a unique bit in the data bus corresponding

to that channel high. For example, the first channel is requested by raising the first bit in the data bus and the *Req* word-line high. Similarly, to request the second channel a source raises the second bit in the data bus. This protocol brings two advantages. Firstly, by making use of the data bus, we avoid the need for additional interconnect. Hence, the size of the fabric which is dominated by the number of wires remains small. A source can also request multiple channels simultaneously. Secondly, in a directory based shared memory system, the home node for data usually maintains a bit vector with 1s corresponding in position to cores that have a shared copy of the data. This bit vector can be directly used in SWIFT to invalidate shared copies of the data. After a channel is used for sending data, the source releases it using a similar protocol. However, this time the *Rel* word-line is used in place of the *Req* word-line. As evident requesting or releasing a channel in SWIFT using this protocol makes use of the data bus and it takes one clock cycle. In one variant of SWIFT, additional dedicated wires can be used for requesting and releasing channels ahead of time to eliminate this single cycle latency. A release operation in SWIFT can also be performed using *Row_Rel* and *Col_Rel* signals without using the data bus. *Row_Rel* is a word-line laid per every row whereas *Col_Rel* is a bit-line available per every column as shown in Fig. 3.6. A source can release all channels held by it by asserting *Row_Rel*. Similarly, a destination can release its channel by asserting *Col_Rel*. *Row_Rel* and *Col_Rel* are asserted in the last cycle of data transfer. Hence, channels can be released without the expense of a clock cycle.

3.4 SWIFT vs conventional fabric

SWIFT uses a distributed arbitration scheme that allows the fabric to scale seamlessly to higher radices. Fig. 3.7 shows how SWIFT's delay and area scale with increasing dimension (number of input/output ports) in comparison with a conventional switch fabric that is optimally designed using standard cells. At higher dimensions

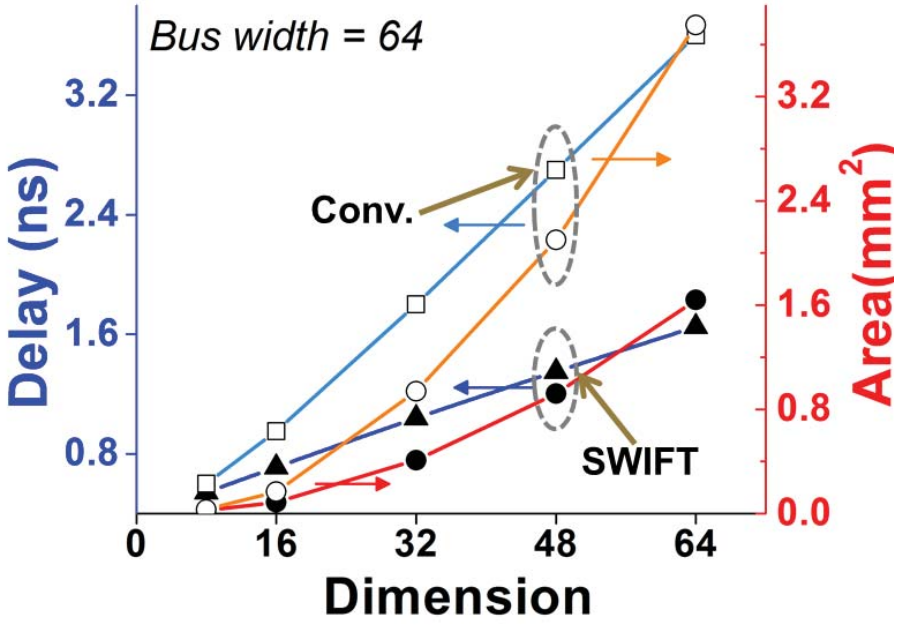

Figure 3.7: Simulated delay and area for SWIFT and conventional fabric

SWIFT consumes 24% less energy while being 2.1× faster. If we scale down SWIFT's operating voltage to match its throughput with conventional fabric operating at nominal supply, energy reduction improves to 69%.

3.5 Test prototype

The test prototype emulates a multi-core system with 32 cores and 32 caches in 65nm bulk CMOS. The cores and caches communicate over a 32 × 32 SWIFT as the interconnect fabric. To save die area, the cores and caches are replaced by traffic generators and signature analyzers respectively. The traffic generators are digitally tuned to produce traffic pattern mimicking real cores. Fig. 3.9 shows the die micrograph with SWIFT and crosspoint layout. The PCB hosting SWIFT test prototype is shown in Fig. 3.10.

Each data bus is 64 bit wide. The 32 × 32 SWIFT has 2048 wires as word-lines coming in and an equal number of bit-lines leaving it. With all these wires laid at

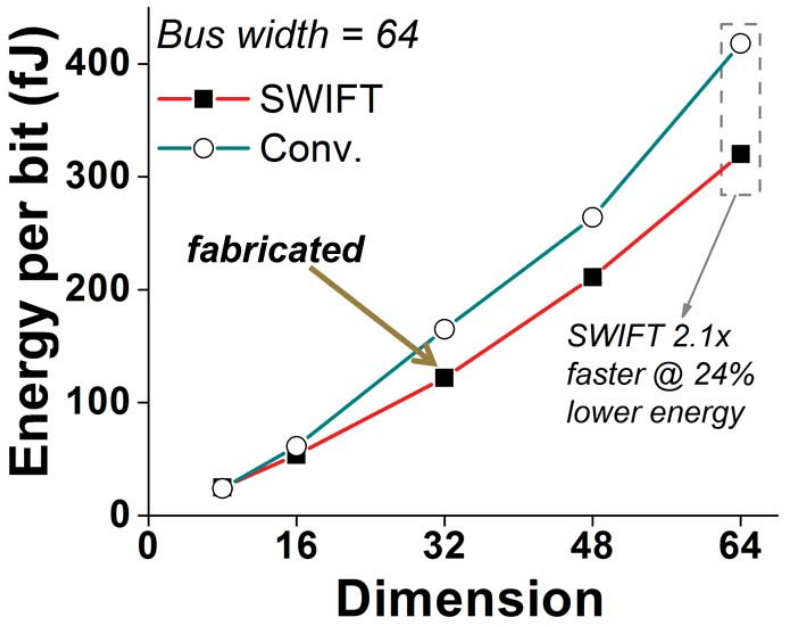

Figure 3.8: Simulated energy efficiency for SWIFT and conventional fabric

2× the minimum allowable spacing, SWIFT spans only 0.35mm^2. SWIFT is made up of 1024 crosspoints. Each crosspoint spans 12.6μm × 20.2μm. The interconnect to logic utilization ratio within each crosspoint is 1:1.

3.6 Circuit implementation

3.6.1 Crosspoint circuit

SWIFT physical layout is very regular in nature and hence an SRAM like design technique can be used for it. Fig. 3.11 shows a typical SWIFT crosspoint schematic. This crosspoint sits at the intersection of the x^{th} column and y^{th} row and is hence referred to as cell(x,y). In cell(x,y) the x^{th} bit in the data bus (in<x> in figure) is used as index while requesting or releasing a channel. In addition, the y^{th} bit-line from the channel (out<y> in figure) is used as the priority line and sampled by the sense amp during arbitration.

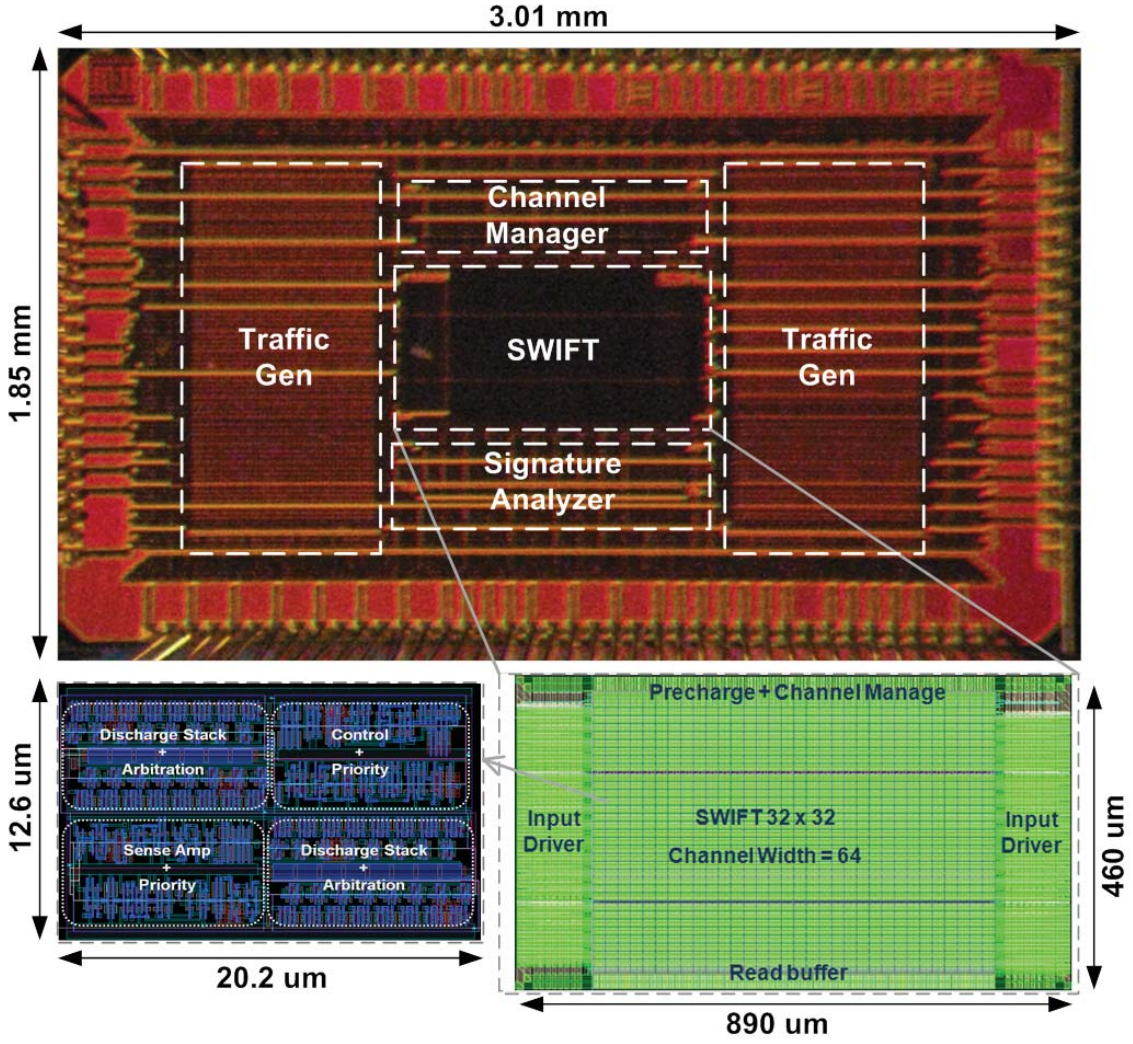

Figure 3.9: SWIFT die photograph with crosspoint layout

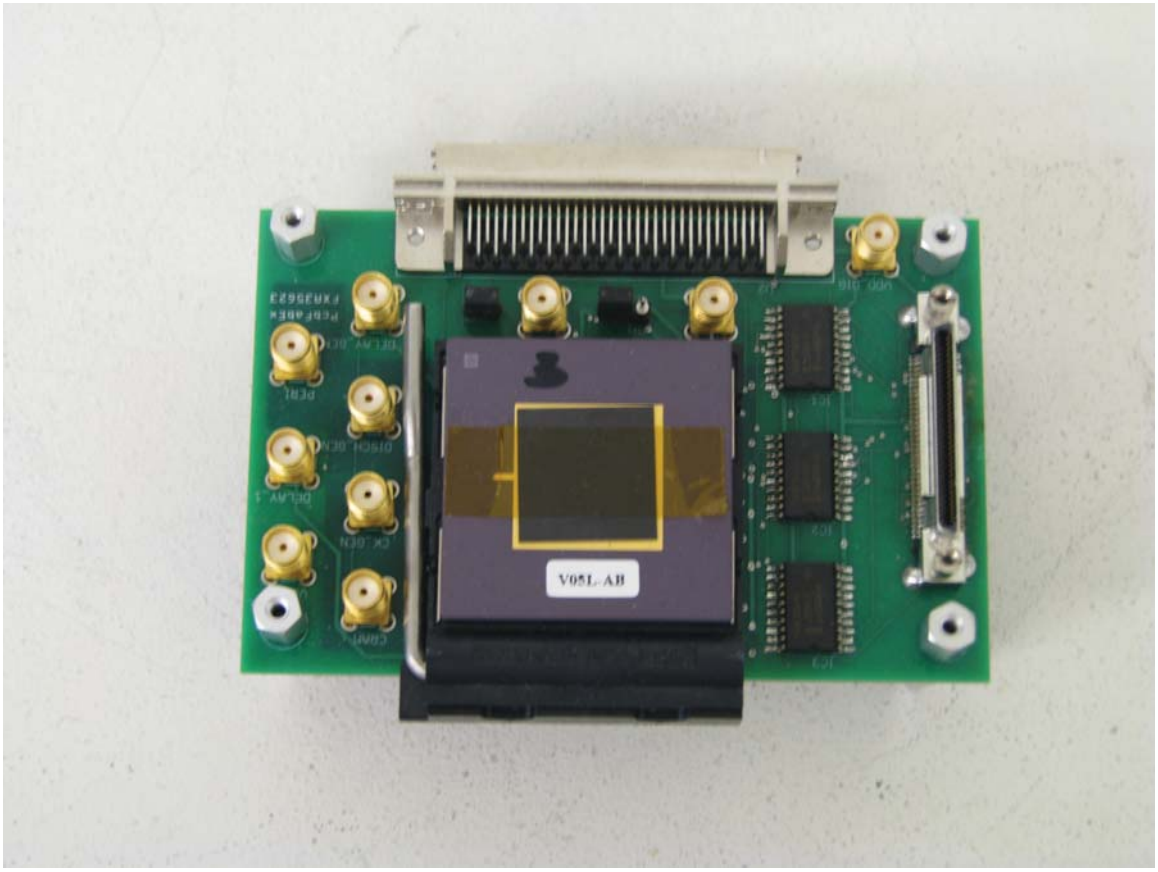

Figure 3.10: PCB hosting SWIFT test prototype

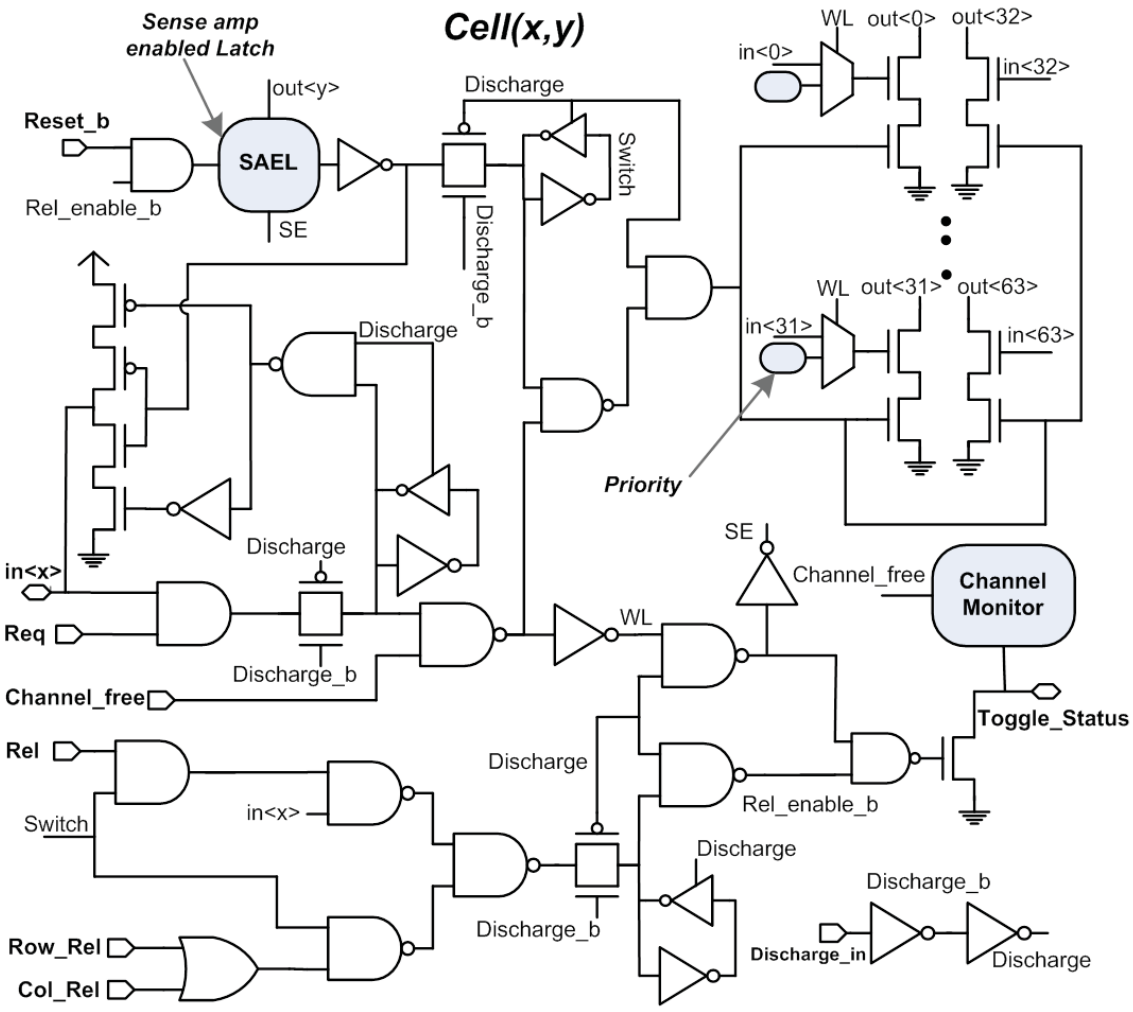

Figure 3.11: SWIFT crosspoint circuit

When *discharge* is low the channel is decoupled from the crosspoint and precharged high. When a *Req or Rel* is asserted, the crosspoint first checks whether the channel is available for arbitration. This information is available from the *channel_free* signal. Depending on whether to transfer data or arbitrate the request, the selection bit for the multiplexers are set. For doing an arbitration, the bit-lines are discharged based on the crosspoints priority. The priority is locally stored at each crosspoint as a bit vector. Each bit in the bit vector uniquely sets the priority of an input in relation to another input. For example let us assume that the top most crosspoint uses the least significant bit-line of the channel as its priority line. In such a case, any other crosspoint with a higher priority (than the top most crosspoint) will have a 1 at the LSB of its priority vector. This ensures that upon requesting the channel the crosspoint can suppress a competing request from the top most crosspoint by discharging its priority line. Similarly, any crosspoint with a lower priority will have a 0 at the LSB of its priority vector.

In the test prototype the data bus and channels are 64bit wide. The SWIFT dimension is 32×32. Hence, only the lower order 32 bit-lines from the channel are used for conflict detection and resolution during arbitration.

During an arbitration cycle each crosspoint selectively discharges*priority lines* corresponding to lower priority inputs as designated in its priority vector. Concurrently, the SAEL in cell(x,y) samples the y^{th} bit-line from the channel as shown in Fig. 3.11. The crosspoint then passes on the result of arbitration onto the indexing bit of the data bus. If the request wins the arbitration, the indexing bit on the data bus stays high. Otherwise it is pulled low. Thus in a single cycle arbitration is done and the acknowledgement is made available to the source. This technique of sending acknowledgement out of the fabric eliminates the need for any additional interconnect. After the crosspoint wins the arbitration it discharges the bit-lines based on the data available on the input bus.

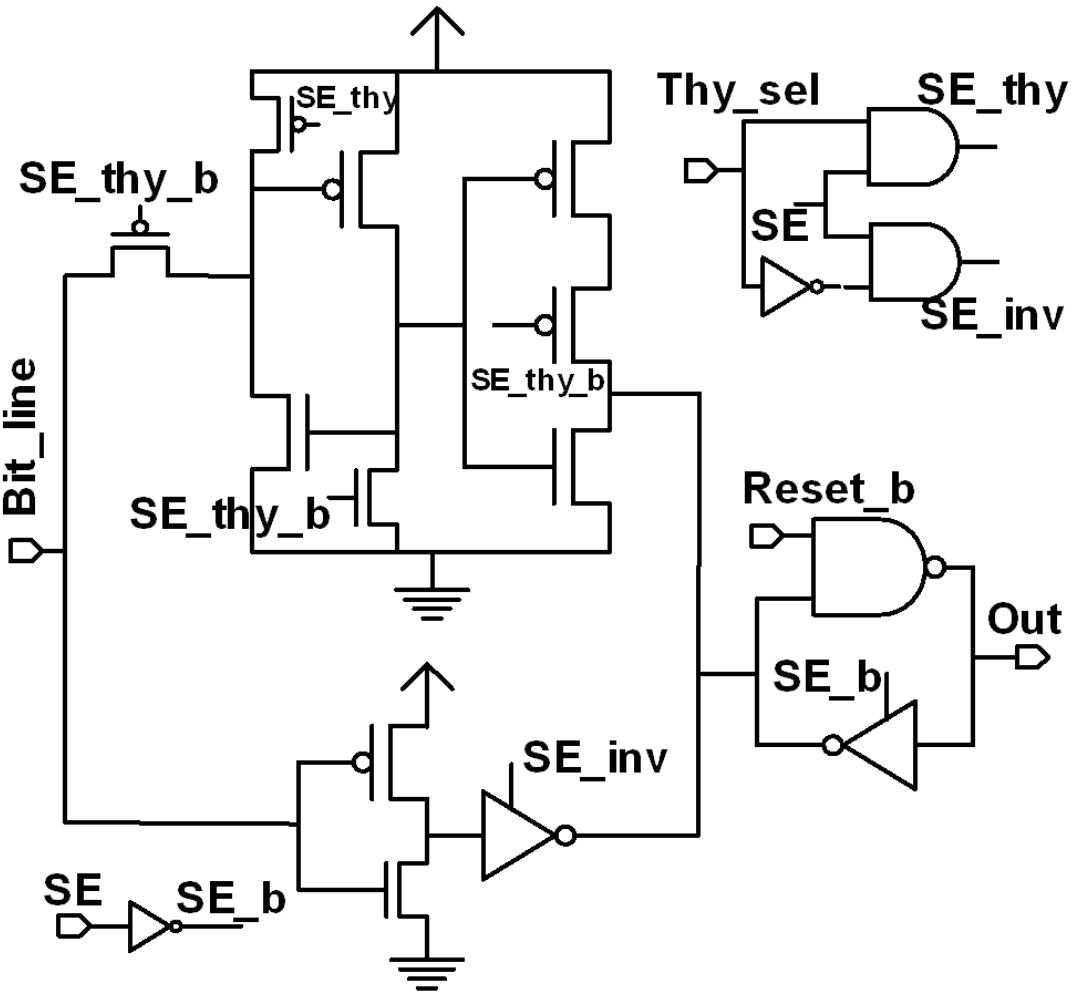

Figure 3.12: Sense amp enabled latch (SAEL) circuit

SWIFT is laid out using a semi custom design flow. A generic crosspoint cell is laid out as a parameterized cell. A skill script parses the generic crosspoint by taking in the x coordinate, y coordinate and the priority vector as the arguments and generates crosspoint specific to each location. These crosspoints are then tiled using a compiler that appropriately sizes the word-line drivers and precharge transistors to generate the SWIFT fabric.

3.6.2 SAEL (Sense Amp Enabled Latch)

The sense amplifier at each crosspoint uses a thyristor based topology in place of a conventional differential amplifier topology for smaller area and better robustness in the face of device mismatch. Fig. 3.12 shows the SAEL circuit. The proposed SAEL is 5× smaller than a conventional sense amp enabled latch. The sense amp can be fired concurrently along with bit-line discharge resulting in a more simplified timing signal generation and distribution. In XRAM, the pre-charge devices in the thyristor were upsized for leakage compensation, degrading performance for functionality at low Vdd. In contrast, SWIFT uses a bypass path with a skewed inverter for low Vdd operation to allow smaller pre-charge devices, improving speed at higher Vdd. The area penalty is small, since SAEL accounts for only 3% of the $254\mu m^2$ crosspoint. The ability to select between the two sensing paths also provides a knob to compensate for process variation at lower Vdd.

The timing diagram in Fig. 3.13 explains SWIFT operation in more detail. In the first clock cycle, during the arbitration phase when the priority line is discharged, the SAEL samples a 0 indicating that the channel was not granted. In the second clock cycle, in a similar arbitration phase, the priority line stays high. The SAEL samples a 1 indicating that the request won the arbitration. In the third clock cycle (after the channel is granted), the bit-line is discharged based on the data.

Only falling transitions are critical, as seen in the timing diagram of Fig. 3.13. This can be leveraged to skew devices to reduce delay. This also eliminates worst case crosstalk because adjacent bit-lines never switch in opposite directions.

3.6.3 Channel status update circuit

Fig. 3.14 shows the precharge and channel status update circuit. Two additional bit-lines called *Toggle_status* and *Channel_free* are added per every column to keep track of whether a channel is available for arbitration. After every successful request

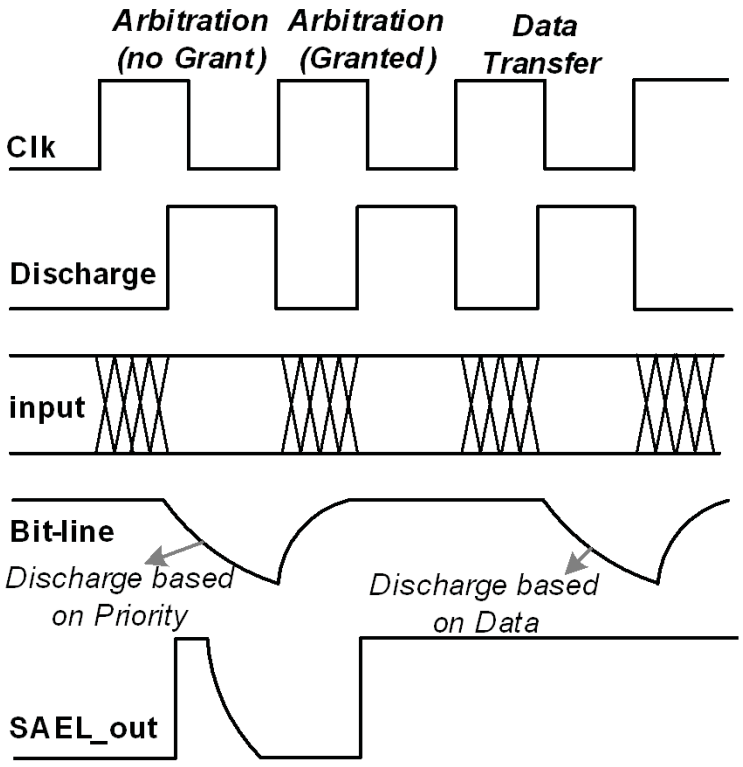

Figure 3.13: SWIFT timing diagram

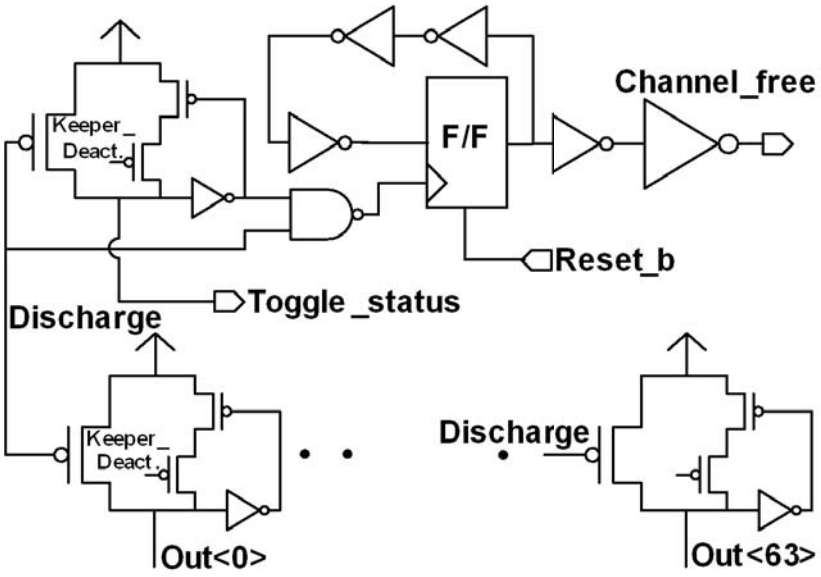

Figure 3.14: Precharge and channel status update circuit

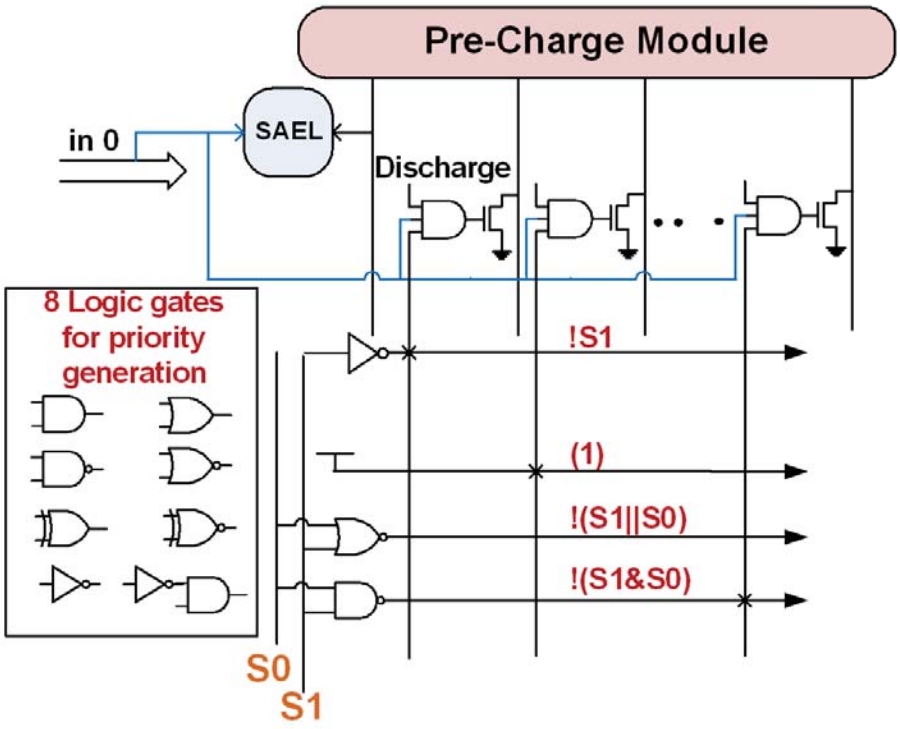

Figure 3.15: Priority vector generation circuit

or release operation (in Fig. 3.11), the crosspoint discharges the *Toggle_status* bit-line. In response to this, the *Channel_free* bit-line is toggled. Also as shown in Fig. 3.14, keepers on the bit-lines can be selectively deactivated to improve performance.

3.6.4 Priority vector generation circuit

For fairness in channel allocation, four sets of priorities are locally generated at each crosspoint and one of them is selectively used for conflict detection and resolution. Every crosspoint has a unique set of 32 bit priority vectors, which if stored as a bit map would require a lot of area. Instead of storing four 32b priority vectors for each of the 1024 crosspoints, eight logic gates, uniquely selected for each crosspoint, indicate which *priority lines* are discharged for each priority as shown in Fig. 3.15. This approach incurs only 3% area penalty over a single priority implementation. This feature is incorporated into the SWIFT skill compiler which automatically selects the

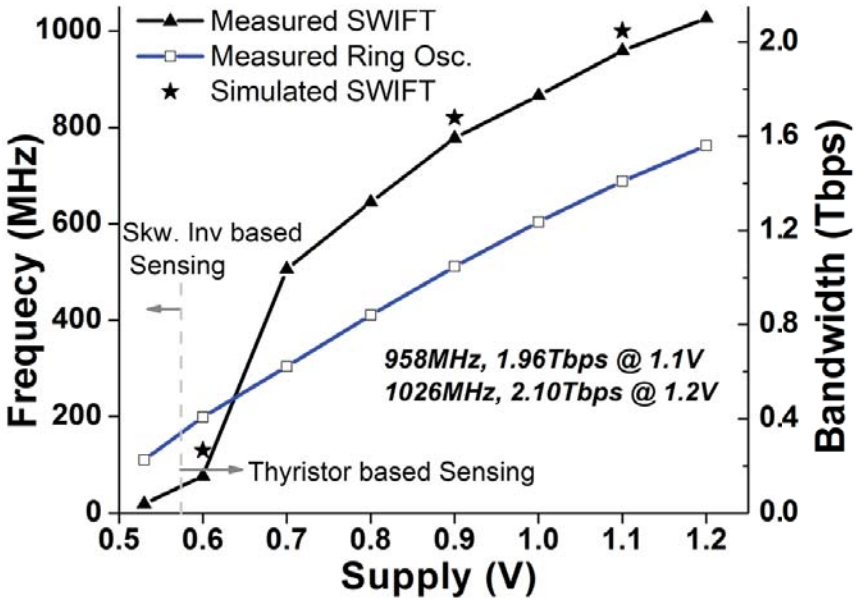

Figure 3.16: Measured SWIFT performance

logic gates and generates appropriate priority vector for each crosspoint.

3.7 Measurement results

Measurement results for 32×32 SWIFT fabricated in 65nm bulk CMOS are shown in Fig. 3.16 and Fig. 3.17. At 1.2V, SWIFT operates at a maximum frequency of 1026MHz. This provides a maximum throughput of 2.1Tb/s. This throughput is measured at single cycle latency. An on chip ring oscillator delay in the same chip is also plotted to compare the frequency vs supply voltage trends. At lower supply voltage SWIFT's bit-line delay degrades faster in comparison with other logic blocks. Hence, a memory array like delay trend is observed as supply voltage is reduced. Fig. 3.17 shows SWIFT's power consumption and energy efficiency (measured as energy per bit) at different operating frequencies. At 20% switching activity on the primary inputs and all 32 channels active, SWIFT consumes 284mW while operating at 1026MHz at 1.2V. This translates to an energy efficiency of 112fJ/bit.

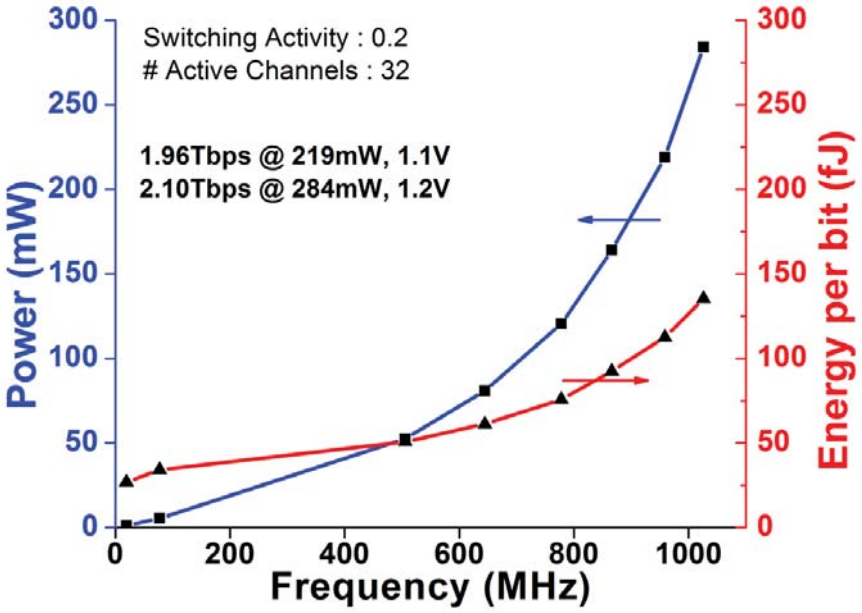

Figure 3.17: Measured SWIFT power and efficiency

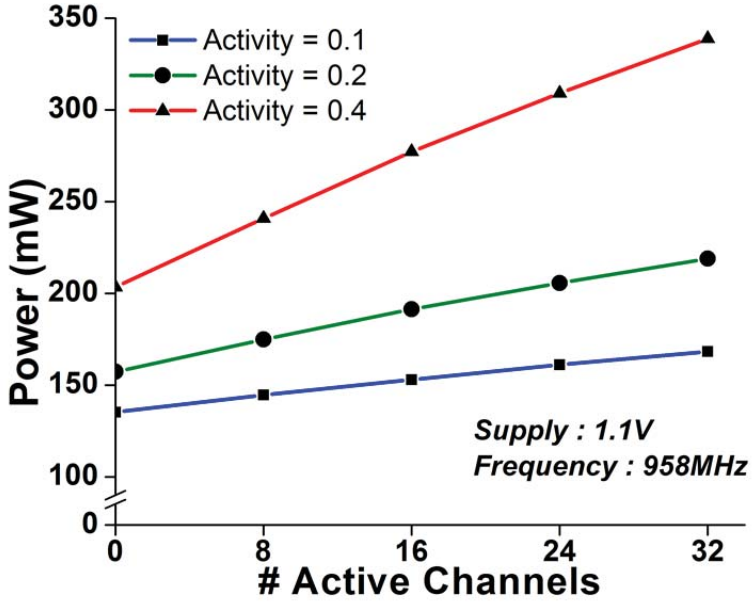

Figure 3.18: Measured SWIFT power with varying number of active channels

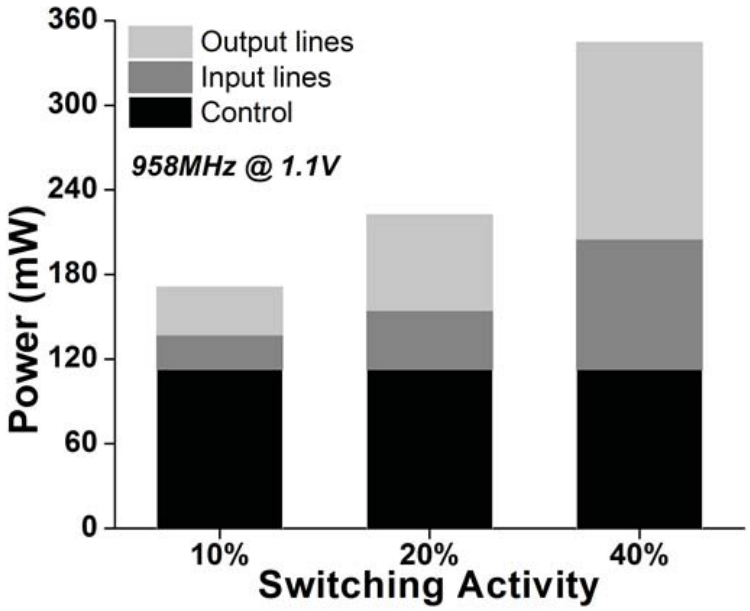

Figure 3.19: Measured SWIFT power breakdown at 1.1V

Fig. 3.18 shows measured SWIFTs power consumption at different switching activities with varying number of active channels. As expected, power increases linearly with switching activity and the number of active channels. Fig. 3.19 and Fig. 3.20 show measured power breakdown within SWIFT. Total SWIFT power is divided into three categories: output line (bit-lines), input lines (word-lines) and control (clock). At low switching activity total power is dominated by clocking and control blocks. At higher switching activity, power breakdown gets more uniform.

Fig. 3.21 shows how the bandwidth for a channel in SWIFT degrades with increasing collision. Collision percentages are set statistically in the test chip by digitally tuning the traffic generators. When requests collide 100% of the time, bandwidth degrades by 44.6% in the worst case. However for typical multi-processor applications, collision rates range from 0% to 10%. At such rates, bandwidth degrades by 2% only. Such low degradation in bandwidth can be attributed to SWIFTs ability to arbitrate in a single cycle at such high radix which takes many cycles in other state of the art

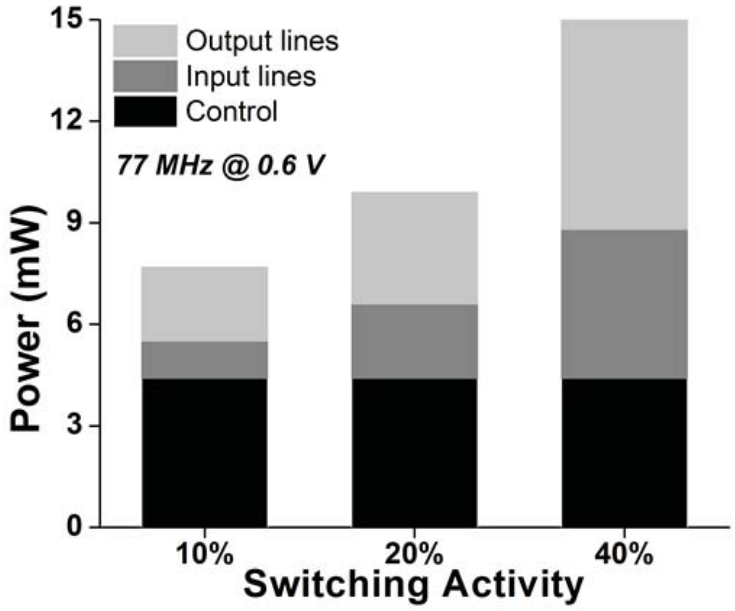

Figure 3.20: Measured SWIFT power breakdown at 0.6V

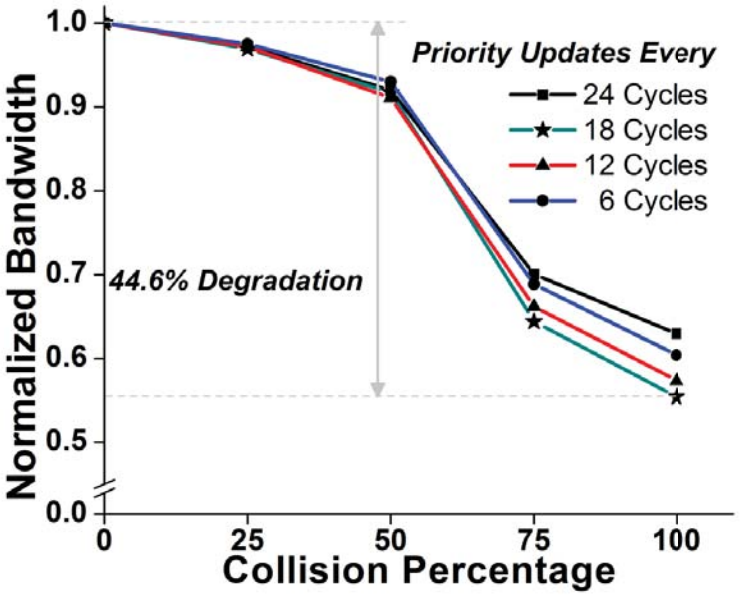

Figure 3.21: Measured bandwidth degradation with increasing collision

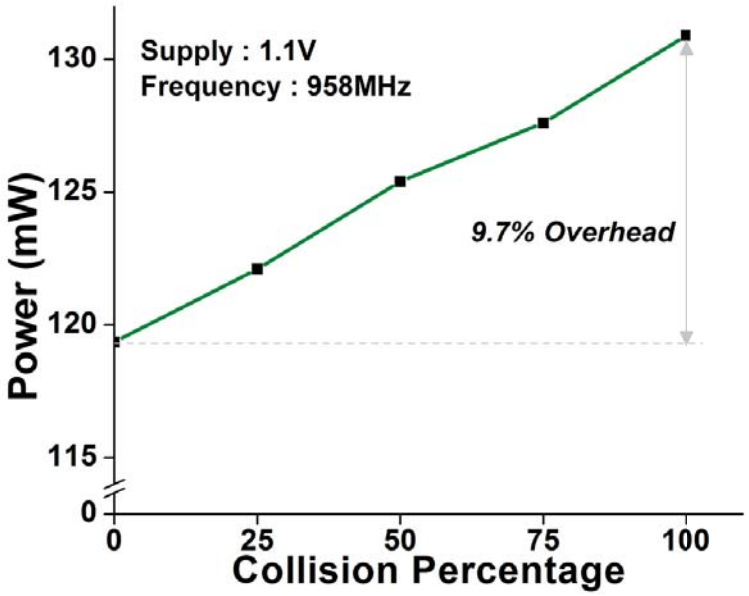

Figure 3.22: Measured power overhead due to arbitration

switch fabrics.

Fig. 3.22 shows measured power overhead with increasing collision. In the worst case, with maximum collision power overhead is 9.7%. However for typical multi-processor applications power overhead due to arbitration stays below 1.5%.

3.8 System level analysis

To study SWIFTs system level architectural implications, we analyzed a multi-core system with three different fabric topologies. Fig. 3.23 shows how SWIFT performs in comparison with a shared bus and an 8×4 mesh in a 32-core system. The shared bus operates at 640MHz in 65nm and has a single cycle latency. The 32×32 SWIFT operates at 1GHz and it takes 3 cycles to get from a source to a destination through the fabric (first cycle is spent in getting to fabric input from source, second cycle is spent through the fabric, and third cycle is spent in getting from fabric output to the destination). The mesh operates at 2GHz and communication latency varies

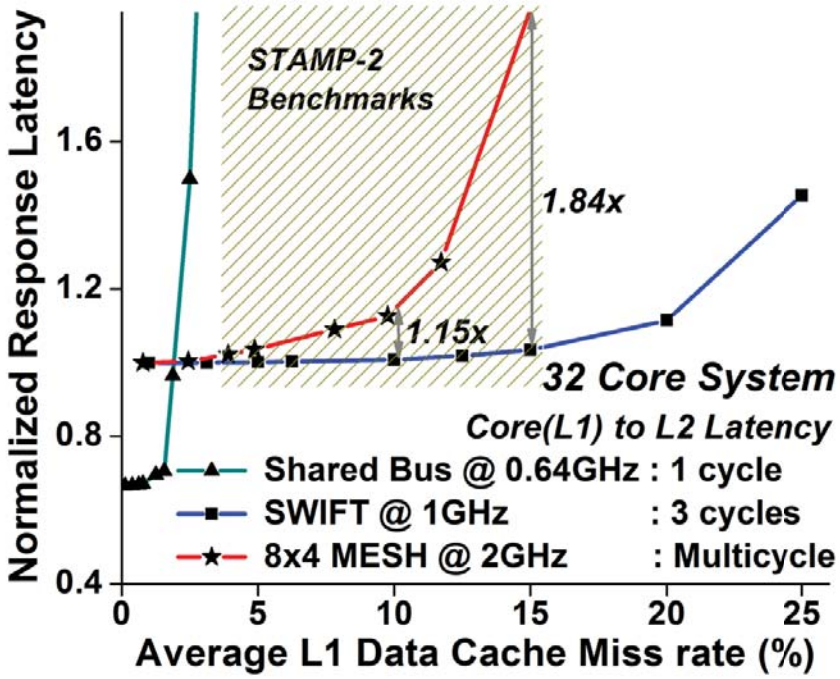

Figure 3.23: Simulated response latency for 32-core SWIFT, shared bus and mesh topologies

from 1 to 11 cycles. As shown, the shared bus saturates with little conflict because of limited bandwidth. SWIFT scales better than mesh with latency improvements ranging from 15% to 84% for STAMP-2 multiprocessor benchmarks owing to superior multicast and conflict resolution ability.

Performance numbers for this comparison were obtained by placing and routing 32 cores and 32 L2-caches with one of the three studied fabrics (bus, mesh, and SWIFT). The interconnect delay was then extracted and the routing overhead studied. For the SWIFT based many core system, the availability of ports at either side of the fabric as datapath-compatible buses (instead of bit interleaved buses) facilitates routing and resulted in the use of less than 6% of total available tracks for connecting cores and caches as shown in Fig. 3.24.

Fig. 3.25 shows the percentage of total SWIFT requests that conflict and involve

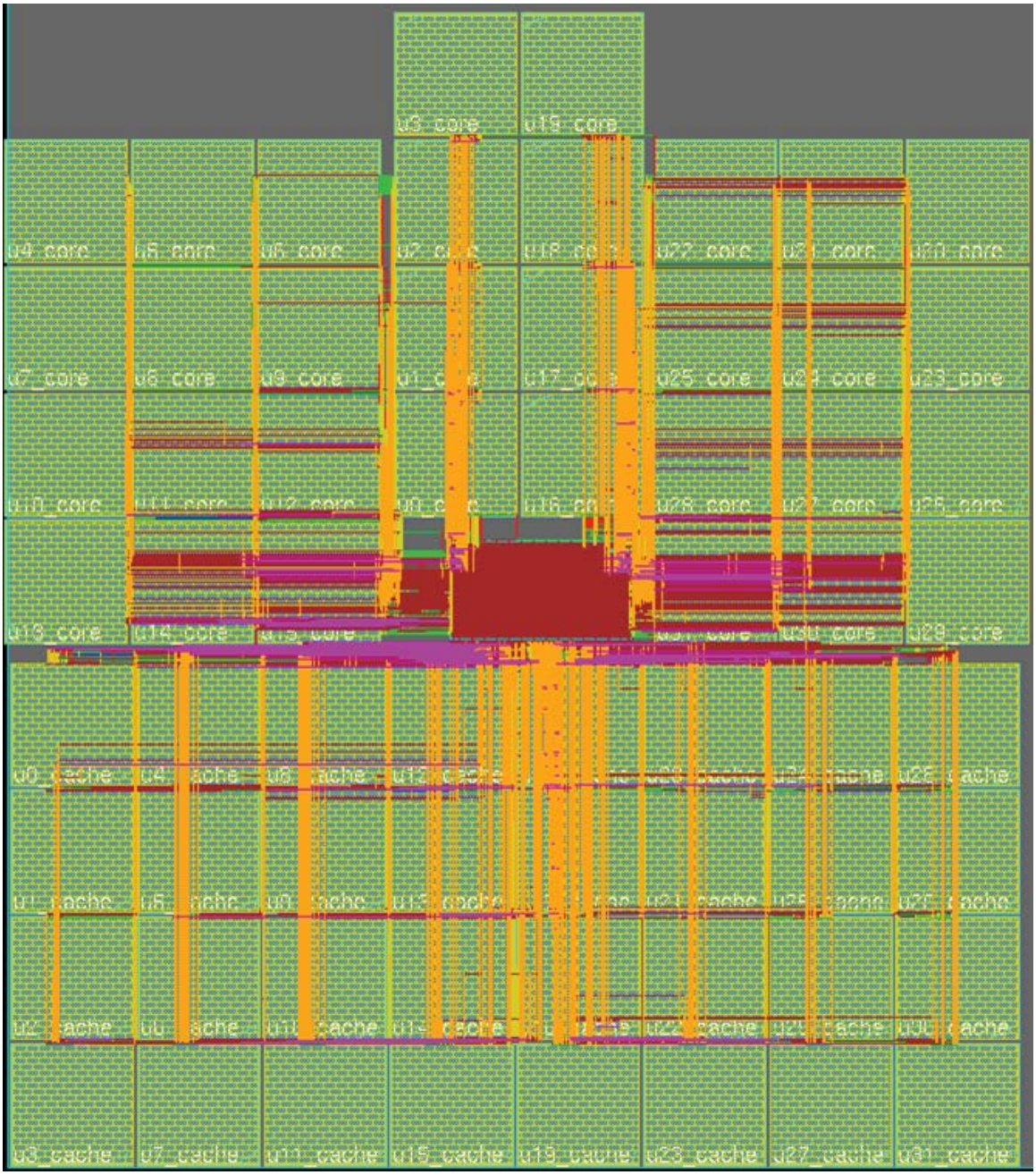

Figure 3.24: System floorplan of 32-core and 32-caches with SWIFT

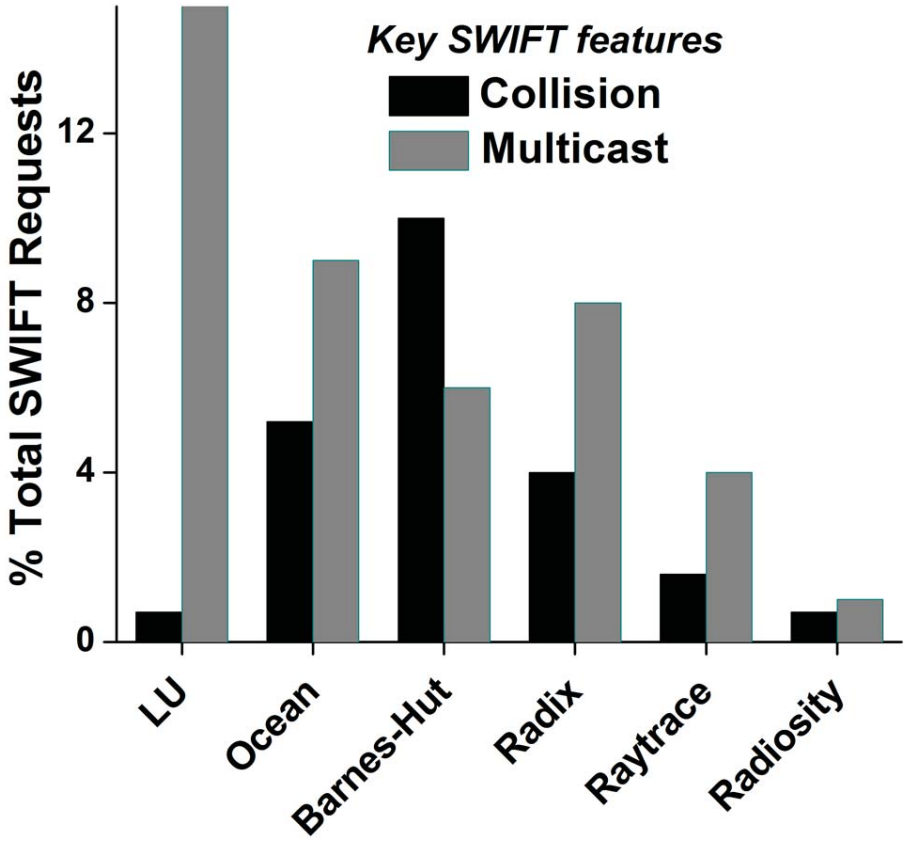

Figure 3.25: Percentage of total SWIFT requests involving collision and multicast

multicast for different benchmarks. As shown, as high as 10% of total requests for some benchmarks can conflict and as high as 15% of all requests can require multicast. A shared bus is inherently good at multicast by its structure. However, only one pair of source and destination can communicate over the shared bus at any instance. Hence, when requests collide it suffers from severe degradation of bandwidth. At the other extreme a mesh handles conflicts well because of distributed control and multiple pairs of sources and destinations can communicate simultaneously. However, because of its blocking nature multicast is pretty time consuming. In contrast, because of its unique fabric architecture SWIFT is well suited for handling both multicast and conflicts.

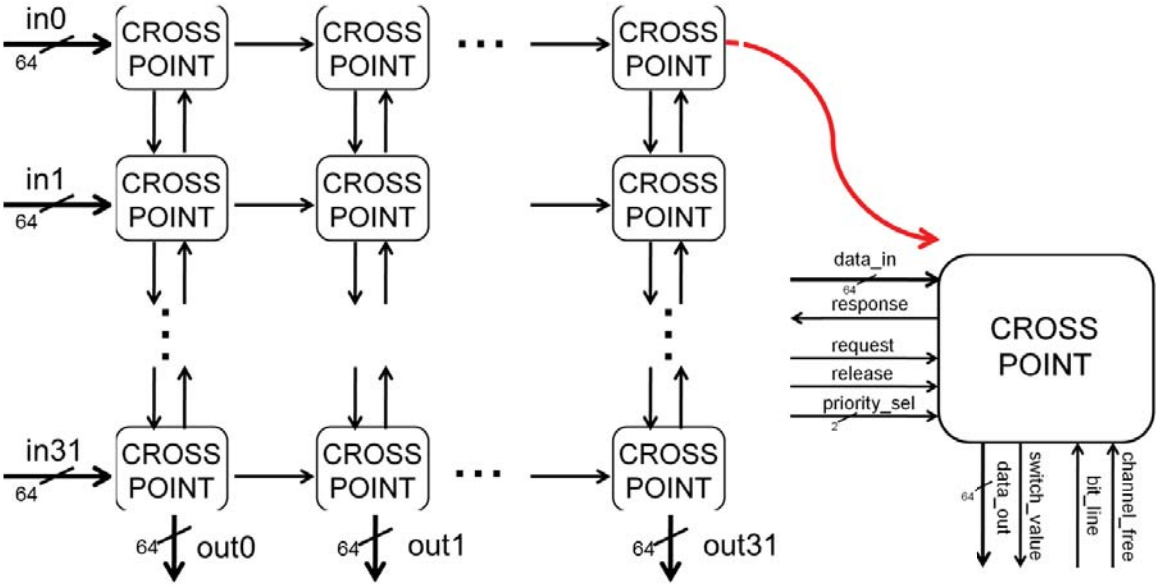

Figure 3.26: SWIFT crosspoint tiling

3.9 Synthesizable SWIFT

An SRAM type semi-custom design approach has been taken to implement the SWIFT fabric discussed in this chapter as shown in Fig. 3.26. This approach has some limitations. It is time consuming and takes several iteration cycles for designing the crosspoints for varying SWIFT radix and bus width. Besides, such an approach is not easily portable to different process technologies. To address this issue, a synthesizable approach is introduced in which the bit-lines in the switch fabrics are replaced by *OR* tree structures. This is shown in Fig. 3.27. The switch fabric designed using this approach still benefits from the unique technique proposed in this chapter that co-optimizes the arbiter and the permutation network by reusing logic and interconnect resources. Instead of having multiple *OR* trees to accomplish fabric functionalities, this approach uses a single *OR* tree. This reduces fabric area and improves performance and energy efficiency.

Simulation studies show that with this synthesizable approach, the arbitration

and priority update techniques still bring significant improvement in fabric latency and power over conventional fabrics. Fig. 3.28 compares 32×32 (64 bit data bus) custom and synthesizable SWIFT fabrics with a conventional farbic that is optimally designed using standard cells. As shown, the unique arbitration technique allows synthesizable SWIFT fabric to run 40% faster and 33% higher energy efficiency over a conventional fabric while spanning 60% less area.

3.10 Summary

In this chapter, we introduced a new fabric topology called SWIFT. SWIFT leverages a novel conflict detection and resolution technique by optimally using already existing interconnect and logic in the fabric meant for data traversal. Fig. 3.29 compares SWIFT with XRAM and another recently proposed switch fabric. SWIFT is 1.9× times energy efficient owing to its small size. It achieves 53% more throughput at 80% lower latency over other fabrics.

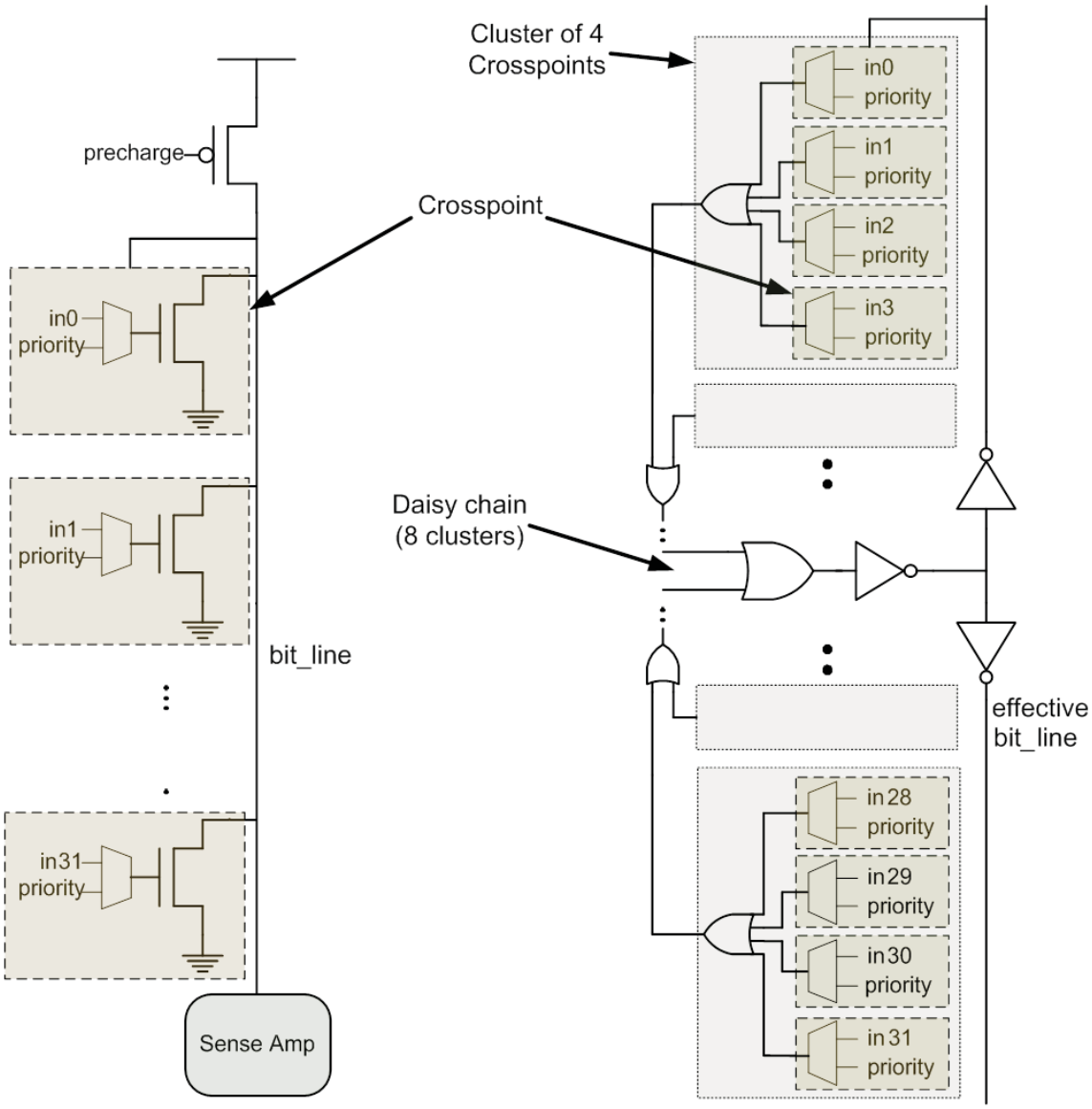

Figure 3.27: Effective bit-line in sythesizable SWIFT

METRIC	Conv. Network	CRAM			
		Std. Cell		Custom	
Frequency (MHz)	625	862	1.4x	1000	1.6x
Area(mm^2)	0.87	0.55	1.6x	0.32	2.7x
Energy(fJ/bit)	157	118.4	1.33x	122.1	1.28x

Figure 3.28: Comparison of custom SWIFT, synthesized SWIFT and conventional farbic

Spec	[58]	XRAM	SWIFT		
Tech.	45nm	65nm	65nm	65nm	65nm
Supply(V)/Temp.(°C)	1.1/50	1.1/27	1.2/27	1.1/27	0.6/27
Size	5 x 5(16B)	128 x 128(2B)	32 x 32(8B)	32 x 32(8B)	32 x 32(8B)
Arbitration	5 to 1	None	32 to 1	32 to 1	32 to 1
Area(mm^2)	0.39	0.75	0.35	0.35	0.35
Max. BW (Tb/s)	1.28	1.07	2.10	1.96	0.157
Frequency (MHz)	2000	530	1025	958	77
Power (mW)	467.5	227	284	219	5.4
Efficiency(Tb/s/W)	2.74	4.71	7.39	8.95	29.1

Figure 3.29: Comparison of SWIFT with some state of the art switch fabrics

CHAPTER IV

SSN : Swizzle Switch Network

4.1 Introduction

Resource management in interconnect fabrics plays an important role in determining overall system performance of multiprocessor systems. This is accomplished in a switch fabric by assigning unique priorities to competing input ports and adaptively updating these based on network traffic and workload. Conventional circuit techniques to implement adaptive priority update techniques for high radix switch fabrics are very expensive in hardware and because of their poor scalability become a bottleneck in improving fabric efficiency at high radices. Usually switch fabrics support a limited number of priority update techniques. The most prominent ones are fixed priority, round robin and least recently granted. However the logic blocks to accomplish these are exclusive. In this chapter, we propose an architecture to accomplish the least recently granted arbitration scheme by reusing already existing logic/interconnect resources in the fabric as shown in Fig. 4.1. Building on that, we also propose novel schemes to accomplish a variety of other arbitration policies with very minimal overhead. The arbitration policies that we incorporate are: 1) Least recently granted (LRG) 2) Most recently granted (MRG) or the greedy algorithm 3) Incremental Round Robin 4) Decremental Round Robin 5) Priority Swap 6) Reverse order 7) Selective LRG 8) Selective MRG. For proof of concept, we have

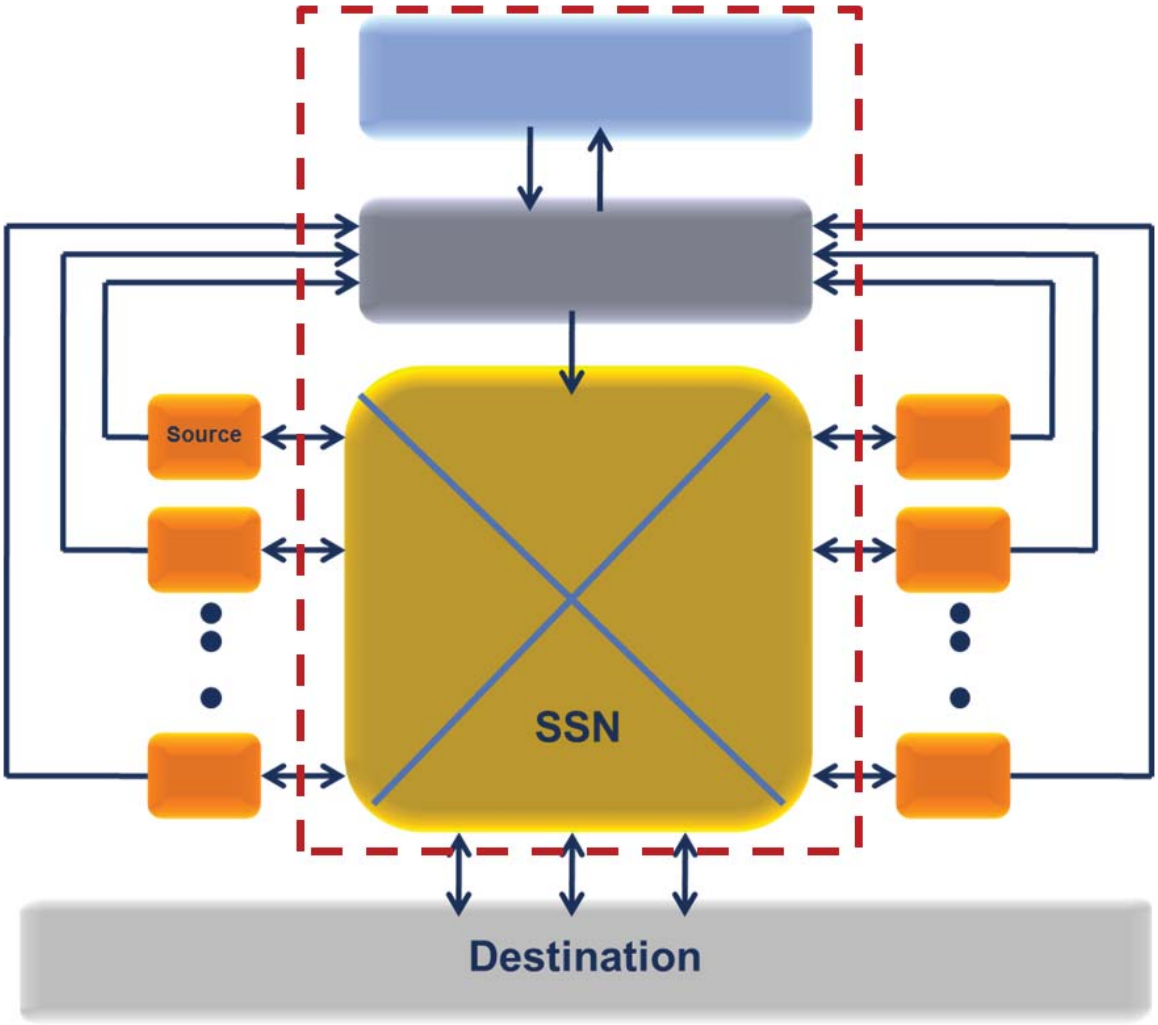

Figure 4.1: SSN: Self arbitrating fabric with adaptive priority update

built a prototype called as the swizzle-switch network (SSN) to demonstrate our proposed technique. SSN is a 64×64 (128b data bus) switch fabric with a throughput of 4.5Tb/s at 3.4Tb/s/W energy efficiency. It spans 4.06mm^2 in 45nm SOI CMOS and achieves a peak efficiency of 7.4Tb/s/W at 0.6V. It features a single cycle least recently granted arbitration technique that re-uses data buses and switching logic. It also integrates a 4-level message based priority arbitration for quality of service and unique bi-directional bit-line repeaters for improved scalability.

4.2 Motivation

High speed and low power routers form the basic building blocks of on-die interconnect fabrics that are critical to overall throughput and energy efficiency of high performance systems. Conventional routers use distinct logic blocks for routing data and handling arbitration [46][58]. At higher radices, connections between these blocks become a bottleneck, limiting router scalability and degrading performance. XRAM and XWIFT merged the data routing fabric with arbitration control, avoiding this bottleneck. However, XRAM relies on centralized control for channel allocation, limiting performance, while SWIFT is restricted to a small set of fixed priorities, rendering input ports prone to starvation. In addition, ever larger CMPs will require continued increases in bandwidth over previous designs. To address these issues, we present a 64×64 single stage swizzle-switch network (SSN) with 128b data buses (8192 input/output wires). SSN can connect any input to any output, including multicast. It has a peak measured throughput of 4.5Tb/s at 1.1V in 45nm SOI CMOS at 25 degrees C. SSNs key features are: 1) A novel, single cycle least recently granted (LRG) priority arbitration technique that re-uses the already present input and output data buses and their drivers and sense amps. 2) An additional 4-level message-based priority arbitration for quality of service (QoS) with 2% logic and 3% wiring overhead. 3) A new bi-directional bit-line repeater that allows the router to scale to >8000 wires. These features result in a compact fabric ($4.06\text{m}m^2$) with throughput gain of 2.1× over [58] at 3.4Tb/s/W efficiency which improves to 7.4Tb/s/W at 600mV.

4.3 SSN architecture

Conventional LRG implementations use controllable delay elements to resolve conflicts [58], which are tuned to change priorities. Large routers require many such delay elements, incurring overhead and probability of meta-stability failures. In contrast,

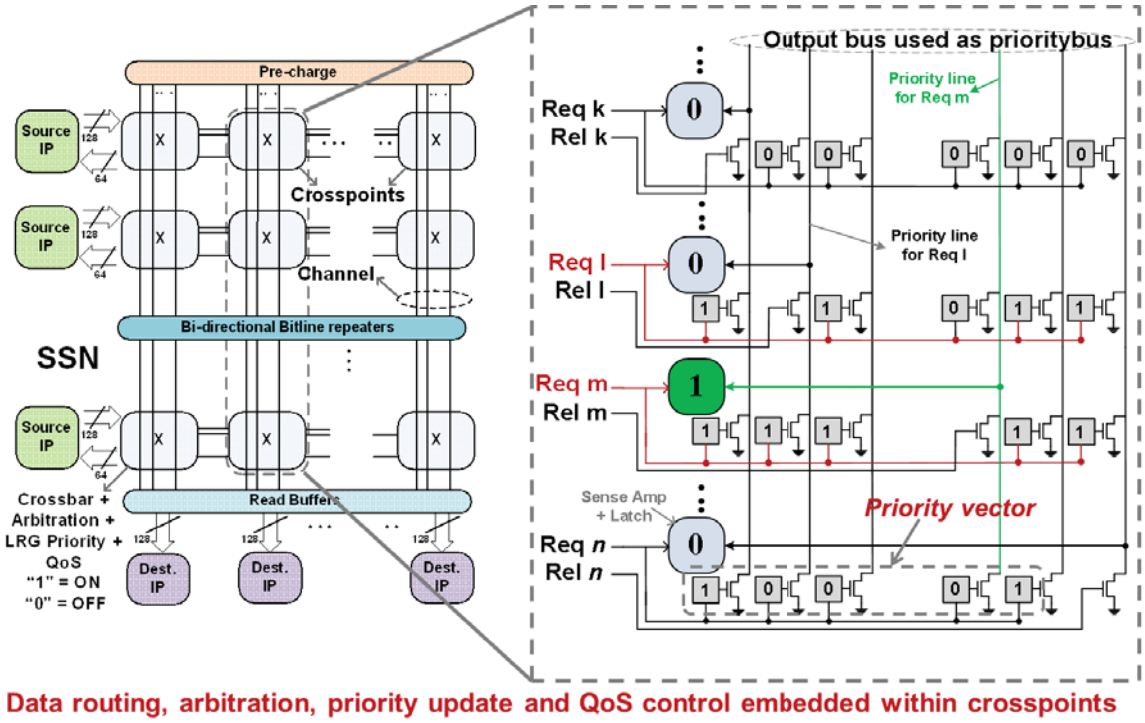

Figure 4.2: SSN fabric architecture

SSN uses a fully static circuit technique that is completely embedded in the data routing fabric by re-using the data-routing bit-lines as *priority lines* for arbitration. The LRG and QoS priorities and the switch configuration are all stored locally at each crosspoint using a novel encoding. Since the data routing fabric is routing limited, this additional logic imposes zero area overhead over a simple switch. Furthermore, as the arbitration logic re-uses the data bit-lines and peripherals, arbitration has the same latency as data transfer and the two scale in tandem with the switch radix.

SSN is a matrix-type fabric as shown in Fig. 4.2 with input buses running horizontally and output buses vertically. When data is routed, the input and output buses transfer data traffic. During arbitration the input bus routes a multi-hot code indicating which output channel(s) are requested by that input, and the output bus is used for conflict detection and arbitration. Each crosspoint stores a connectivity status bit indicating whether the input bus was granted access to the output channel. A 63-bit

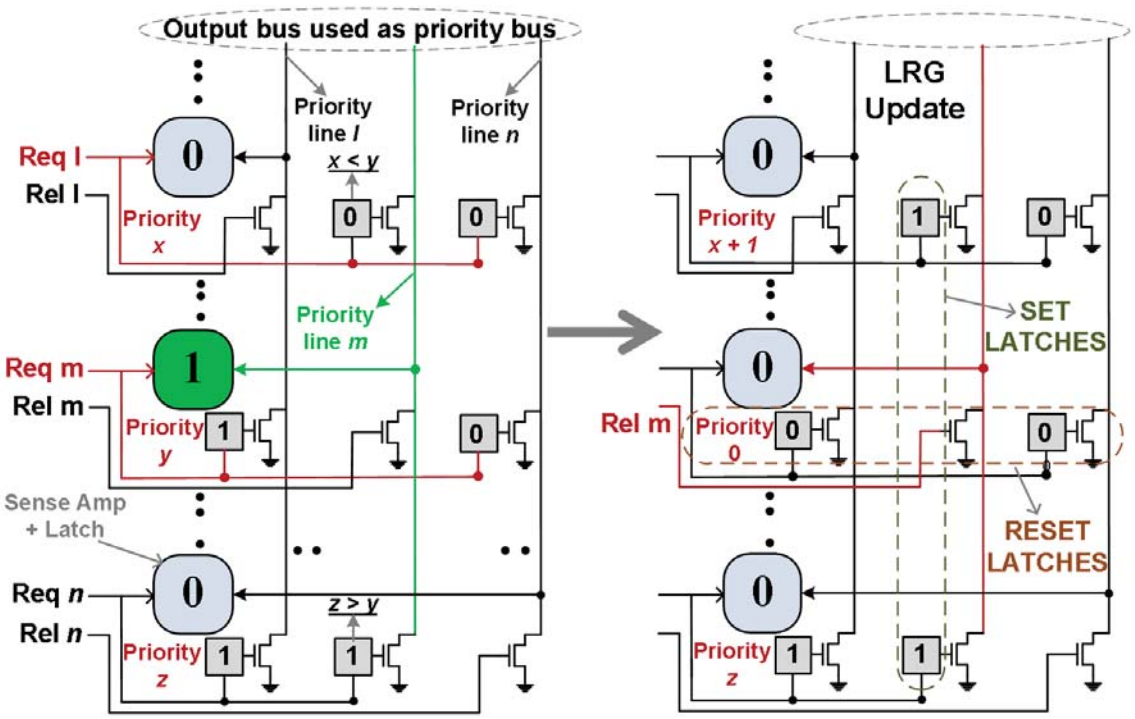

Figure 4.3: LRG update technique

priority vector is also stored to represent the priority of the input bus with respect to all other inputs for that output bus. Fig. 4.2 (right) shows the priority vector at each crosspoint in a blow-up of a single output channel. Each input bus is assigned a unique bit-line from the channel as its priority line which, if high, indicates it as the winner in a particular arbitration cycle. Similarly, each bit in the priority vector at a crosspoint corresponds with a *priority line* (bit-line) and indicates whether the input bus at that crosspoint has higher or lower priority than the input bus associated with the *priority line*. For instance, in Fig. 4.2 *priority line* m corresponds to input bus m while the m-*th* priority bit of bus n is a 1, indicating that n has higher priority than m. When input n requests the output channel this high bit results in the discharge of priority line m, suppressing access by input m. In contrast, input l stores a 0 at its m-*th* priority bit and hence does not suppress an access request from input m, meaning that l has lower priority that input m.

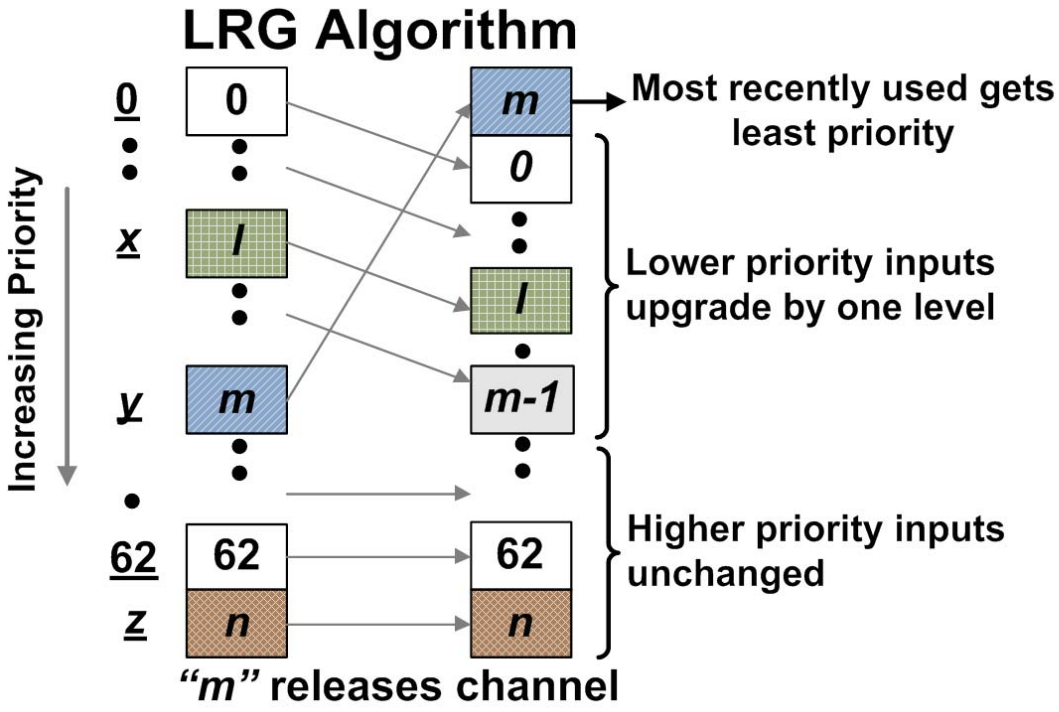

Figure 4.4: LRG algorithm

Priority vectors need to be set consistently and indicate the same priority order. In Fig. 4.3, the 0 at bit m of input l must be mirrored with a 1 at bit l of input m. Furthermore, the priority bits need to be correctly updated after each arbitration cycle to implement LRG policy. We propose a new, simple mechanism to accomplish this. In Fig. 4.3, inputs l and m request the output channel in an arbitration cycle. Input m wins owing to its higher priority and its connectivity status bit is set to 1. After data transfer, input m releases its channel during a channel release cycle. In this cycle, input m first resets all its priority bits. This guarantees that m now has lowest priority, as required by the LRG algorithm. At the same time, input m also lowers its *priority line* m, which is a signal to other crosspoints in the output channel to set their *m-th* priority bit. This ensures that all other input buses now have higher priority than m. Input buses with higher priority than m remain unchanged and only inputs with lower priority than input m are increased in their priority by exactly one

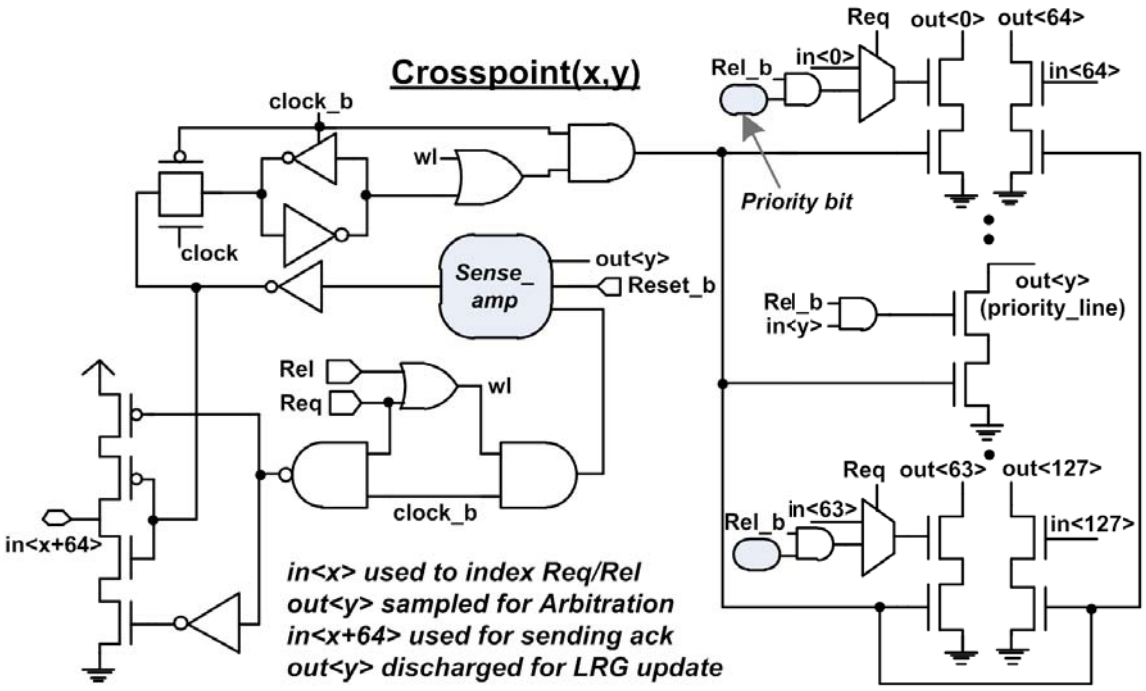

Figure 4.5: SSN crosspoint circuit

level. This simple and fast update mechanism provably guarantees both consistency of all priority vectors and correct implementation of the LRG arbitration scheme, which enables efficient and deadlock-free routing [60]. The LRG update for this case is shown in Fig. 4.4.

4.4 Circuit implementation

4.4.1 Crosspoint circuit

Fig. 4.5 shows the SSN crosspoint circuit and Fig. 4.6 shows the priority storage latch. *Load_priority_b* is an additional bit-line provided per channel that is discharged during the release cycle. This triggers the priority update mechanism. During a request/release cycle the channels are indexed using the lower 64 bits from the input bus. Crosspoints send acknowledgement over the upper 64 bits.

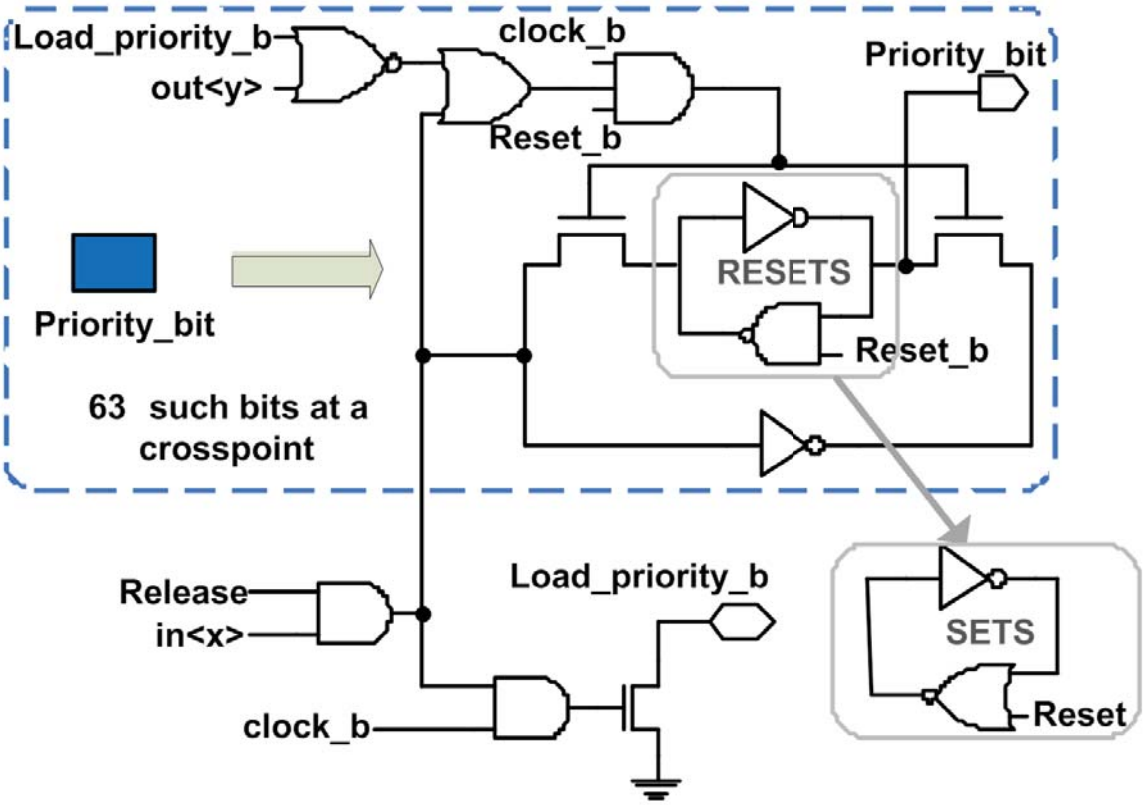

Figure 4.6: Priority storage circuit

4.4.2 Message based quality of service arbitration

SSN also features a 4-level message-based QoS arbitration technique that allows only input buses with the highest message priority to arbitrate for the channel. This is shown in Fig. 4.7. Every input bus is supplemented with two additional word-lines to carry a priority indicating the preference with which the message is requested to be routed. The 2-bit message priority is decoded into a 4-bit thermometer code at the crosspoint, which is used to selectively discharge priority bit-lines comprising the QoS priority bus. A multiplexer samples one of those priority bit-lines using its own message priority and the input bus progresses to the LRG arbitration cycle if the monitored priority bit is not discharged. Using separate wires for QoS arbitration incurs 3% area overhead. However, the additional QoS arbitration cycle can be overlapped with the prior routing operation for the output bus, avoiding a latency penalty. Fig. 4.8 shows the QoS arbitration algorithm. In the first cycle, only requests with the highest message priority are selected. Following this in the second cycle only these filtered requets participate in the LRG based arbitration process.

4.4.3 Self regenerating bit-line repeater

The SSN features 8448 word-lines and 8576 bit-lines spread across 4096 crosspoints. The integration of the LRG and QoS control within this fabric with low overhead greatly improves SSN scalability to realize a fabric of large size. In addition, new bi-directional repeaters are used for bit-lines that use a regenerative sensing element to improve delay despite high slew rates on long bit-lines. Fig. 4.9 shows the repeater schematic.

The proposed repeater uses a thyristor element to detect and amplify a transition on the bit-line. Once a transition is detected the repeater enters a self regeneration mode where it decouples itself from the slow transitioning bit-line. This allows the internal nodes in the thyristor to switch faster and reduces delay. SSN fabric operation

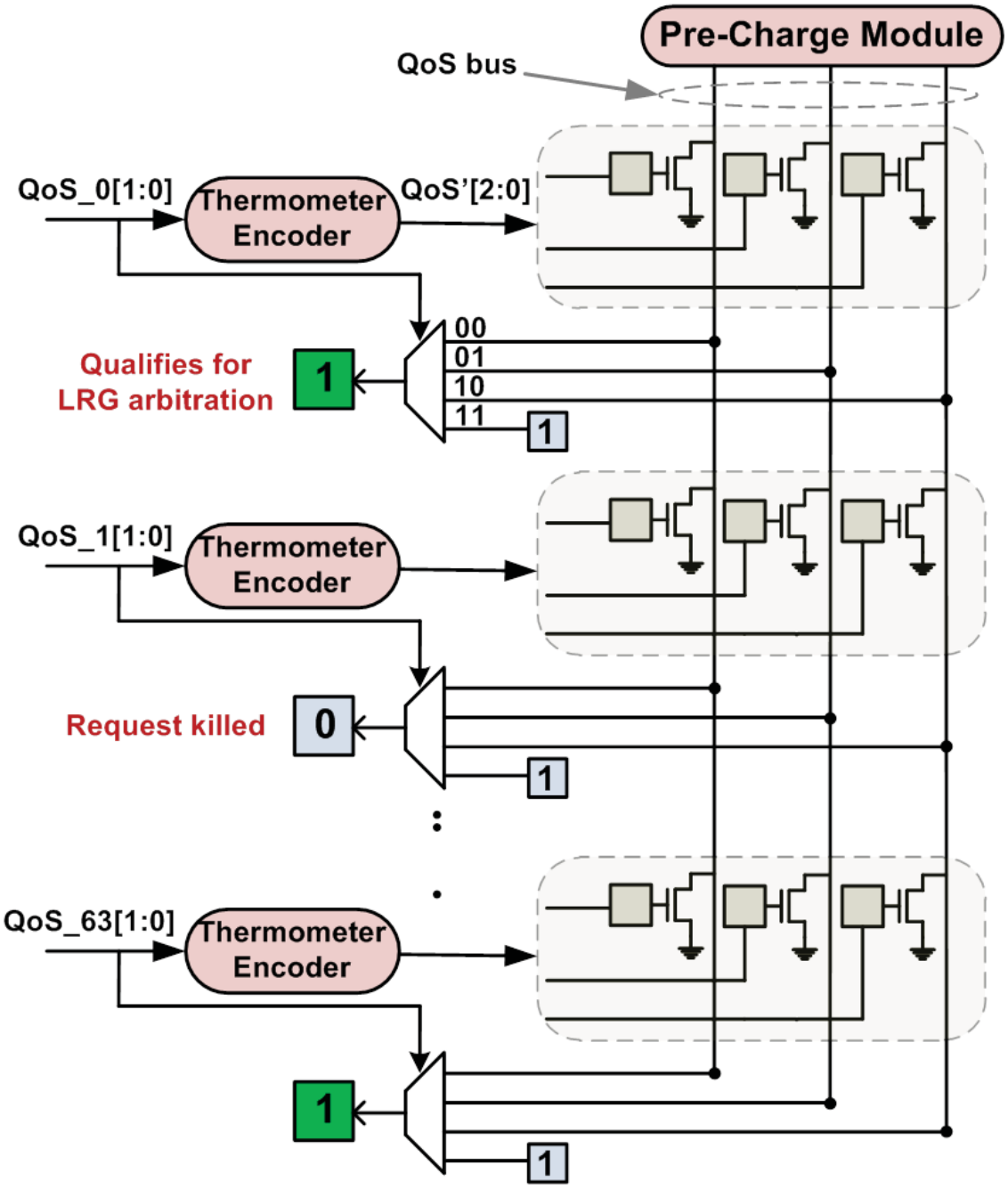

Figure 4.7: Message based quality of service arbitration technique

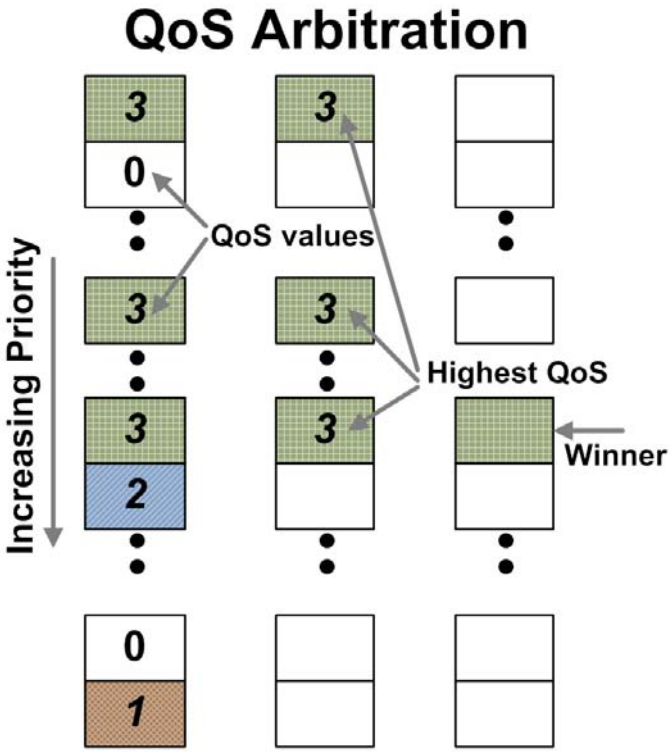

Figure 4.8: QoS arbitration algorithm

in different modes is explained in more detail in the timing diagram shown in Fig. 4.10.

4.5 Design choices for scalability

The regeneration and self-decoupling mechanism improves bit-line delay by 32% and allows for a 50% smaller bit-line driver compared to a conventional repeater (Fig. 4.11, simulated). Simulated fabric latency with increasing SSN size shows 1.6× performance improvement over an SSN with un-repeated bit-lines due to the near-linear latency increase with radix size rather than quadratic dependency without repeaters as shown in Fig. 4.13. Bit-lines are pre-charged within every 16×16 SSN macro. This improves pre-charge time by 59% over a similar sized lumped driver and results in more uniform current drawn from the power grid. Bit-line delay degrades more rapidly than word-line delay under voltage scaling. Hence, the bit-cell aspect

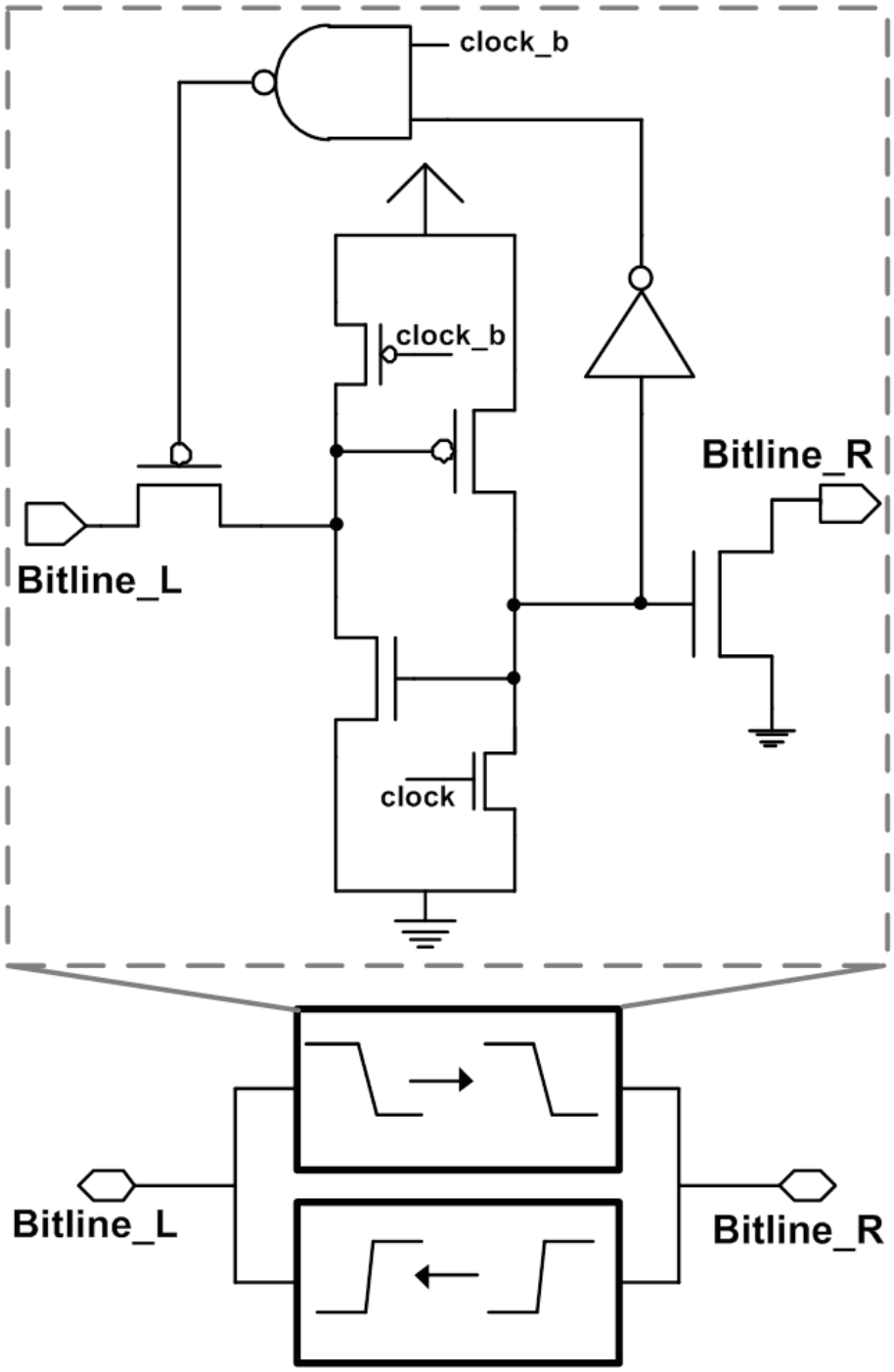

Figure 4.9: Self regenerating bit-line repeater

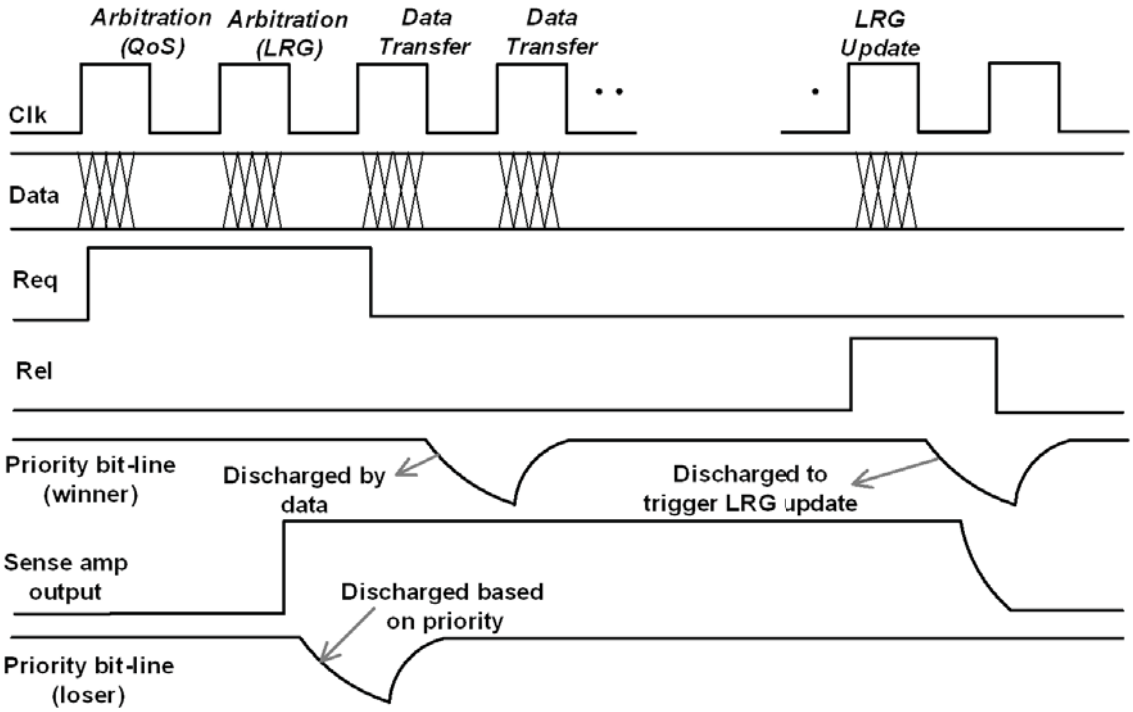

Figure 4.10: SSN timing diagram

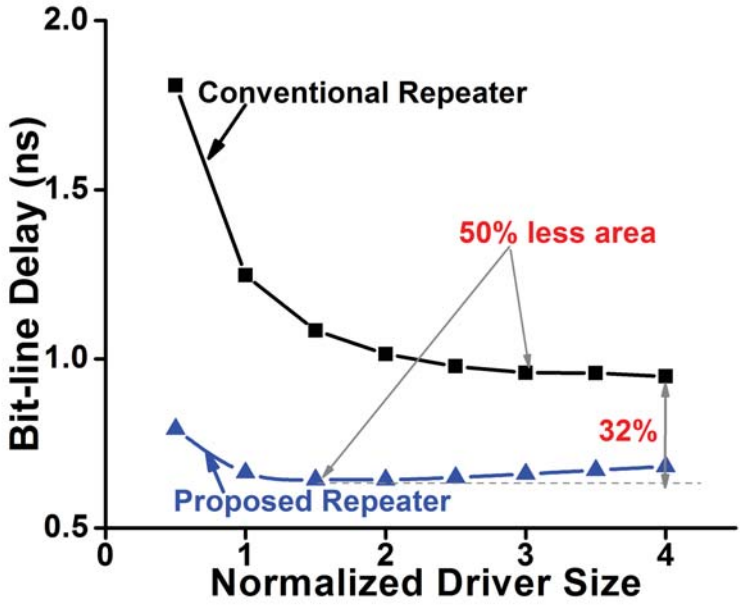

Figure 4.11: Simulated bit-line delay with conventional and proposed repeater

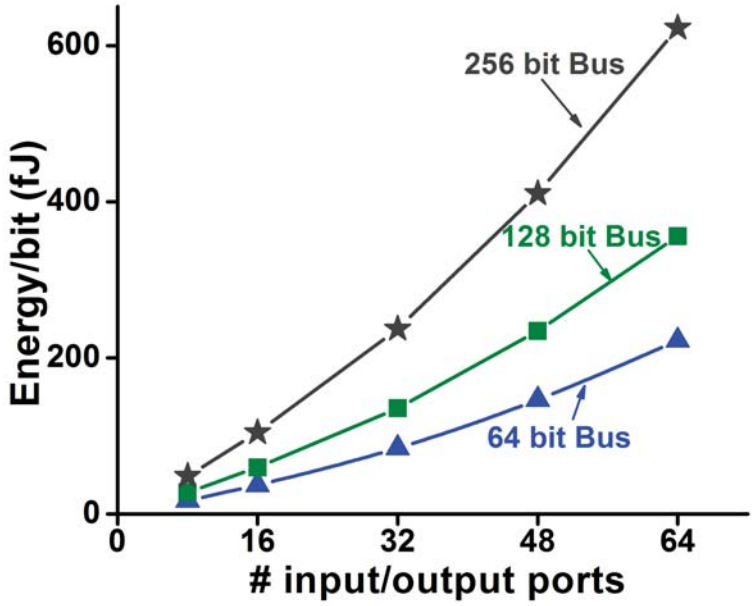

Figure 4.12: Simulated SSN energy efficiency with increasing radix

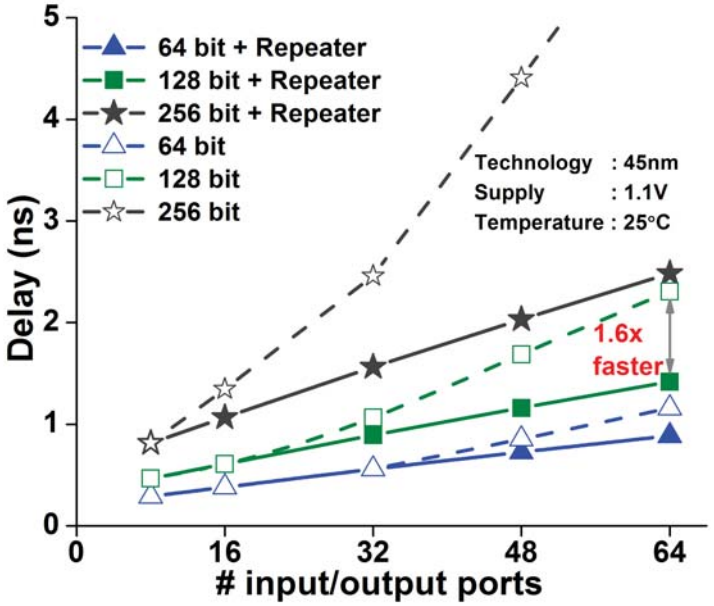

Figure 4.13: Simulated SSN delay with and without proposed repeaters

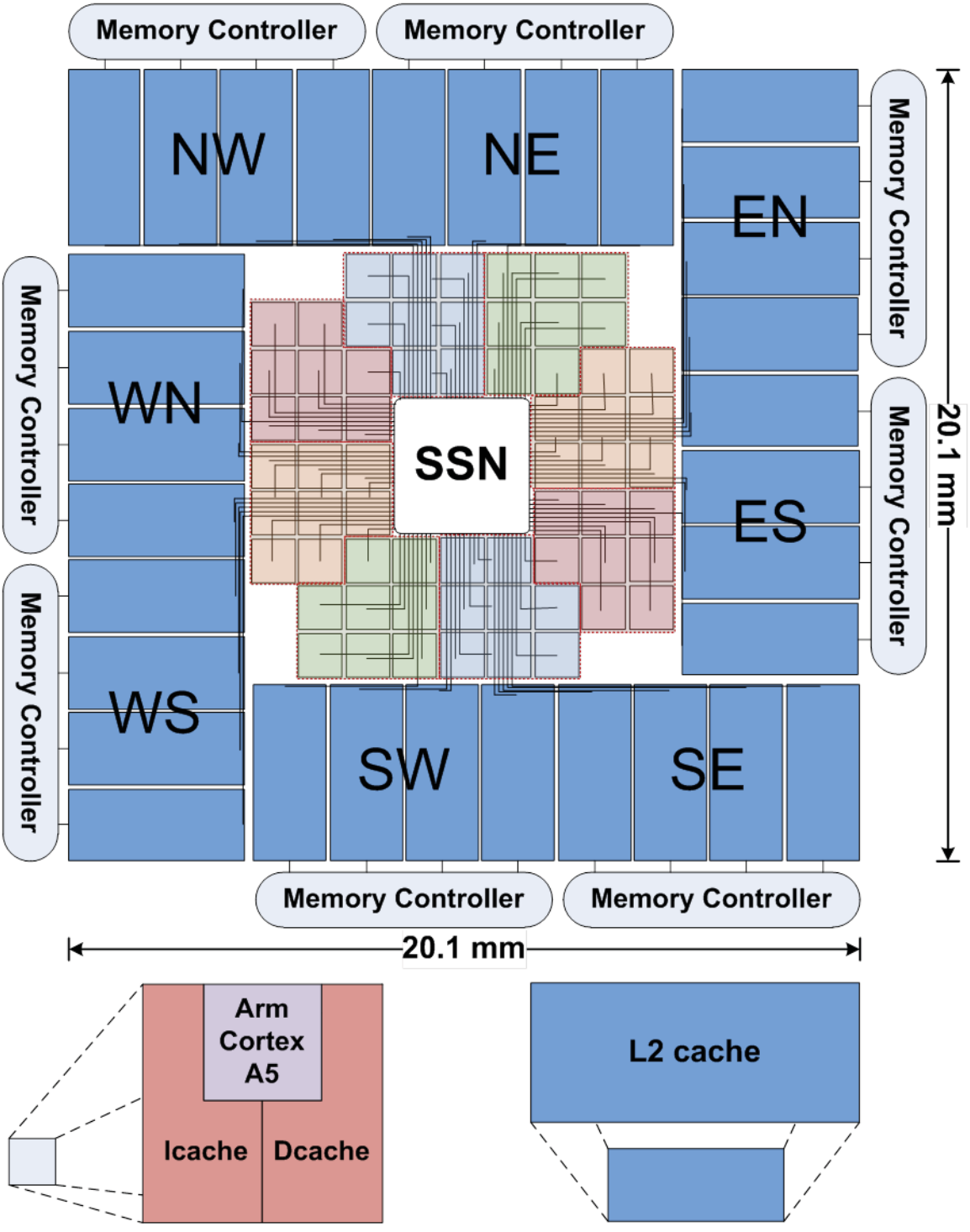

Figure 4.14: 64-core system floorplan with SSN as the interconnect fabric

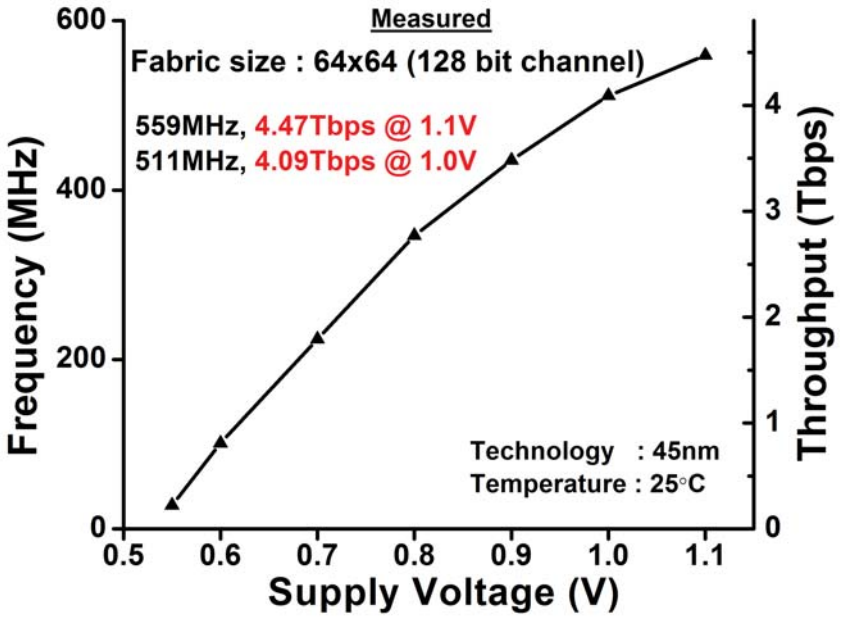

Figure 4.15: Measured SSN performance

ratio (1:0.73) is chosen to shorten bit-lines, improving fabric latency at low Vdd. Fine grain clock gating reduces clock power by 94% at each crosspoint. A crosspoint is clocked only if its connectivity status is ON, a request is asserted, or an LRG priority update occurs. These events are registered in the positive clock phase, allowing gating of the negative (active) phase with 2.3% delay penalty. Adjacent SSN input ports are driven from opposite directions, reducing routing congestion and local Ldi/dt voltage drop when repeaters on the 2.5mm long word-lines switch.

4.6 Test prototype

Fig. 4.17 shows die micrograph of the test prototype. SSN and crosspoint layouts are also shown alongside. System specifications are listed in Fig. 4.18. Fig. 4.19 shows the PCB board hosting SSN test prototype.

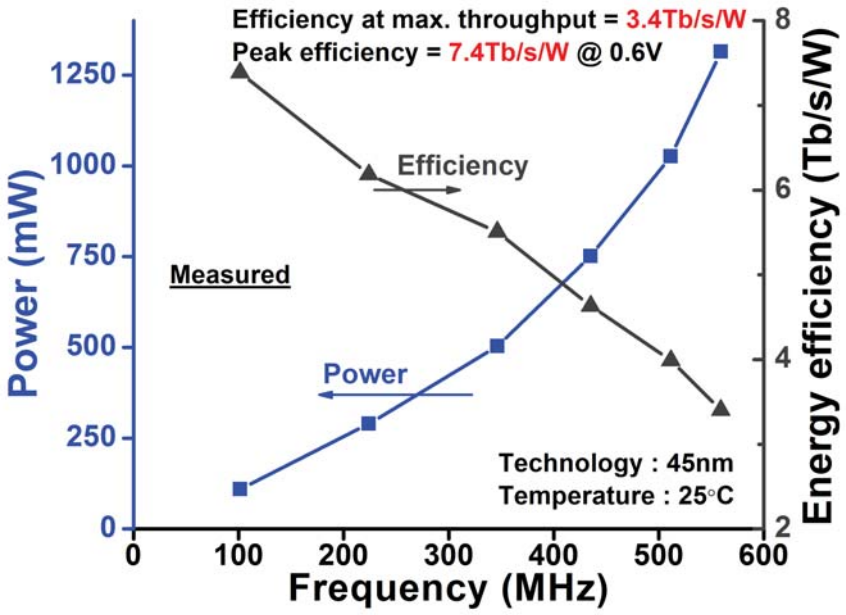

Figure 4.16: Measured SSN power and energy efficiency

4.7 Results

Measured performance and power for a 64 × 64 SSN in 45nm SOI CMOS are shown in Fig. 4.15 and Fig. 4.16 respectively. The SSN achieves 4.5Tb/s at 1.1V with an efficiency of 3.4Tb/s/W at 20% switching activity with random traffic. SSN is fully functional down to 550mV with a measured peak efficiency of 7.4Tb/s/W at 0.6V.

Comparison results of SSN with some of the state of art switch fabrics (fabricated recently) are shown in Fig. 4.20 and Fig. 4.21. SSN achieves 10% higher throughput at 3.7× higher energy efficiency and 3× lower latency, making it very suitable for networking in large scale multi-processor systems.

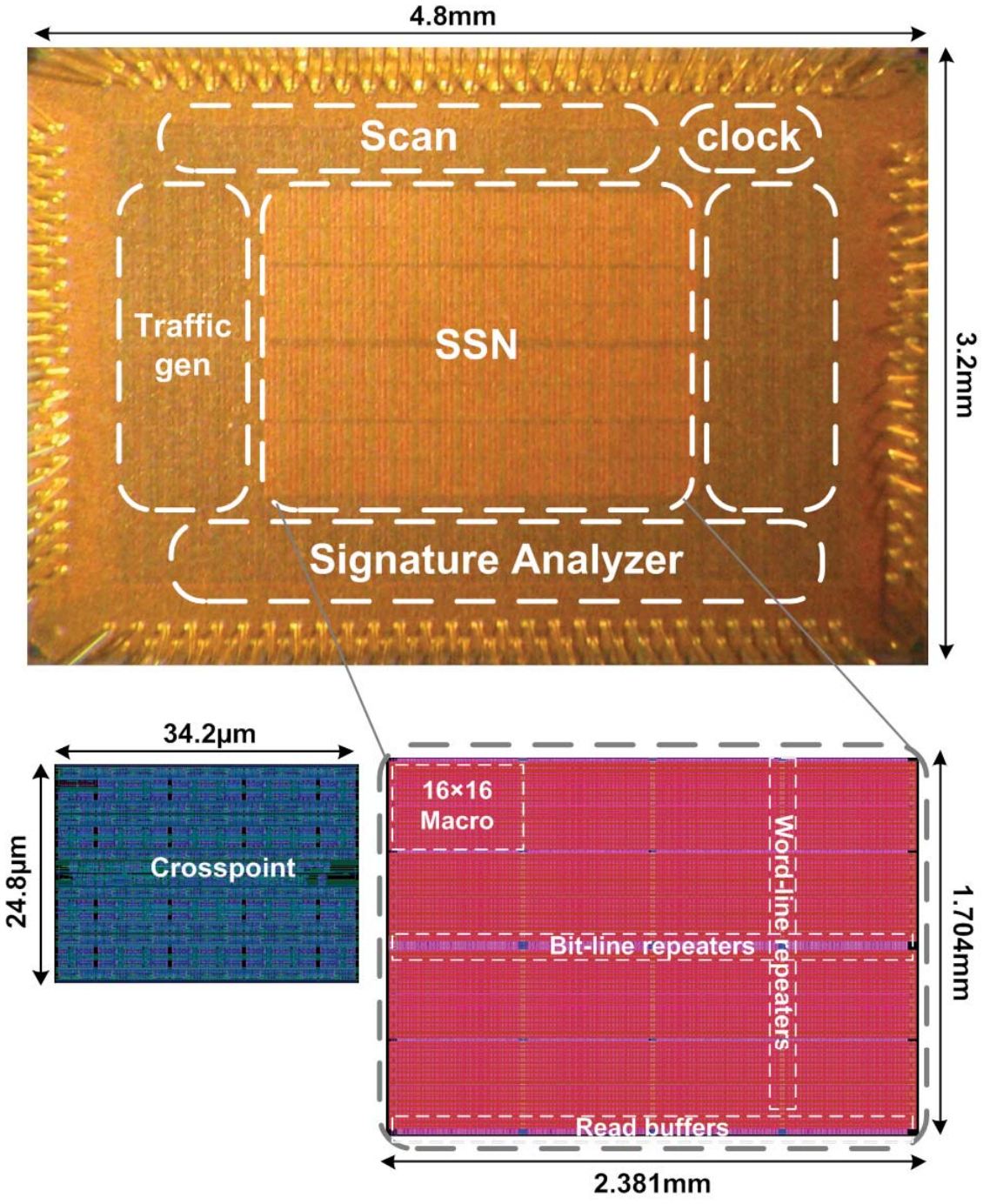

Figure 4.17: SSN die photograph with crosspoint layout

Process	45nm SOI CMOS 12metal interconnect
Die area	15.6mm^2
Fabric area, Transistor count, # Data wires	4.06mm^2, 6.95M, 8192
Throughput, Frequency	4.47Tb/s @ 1.1V, 559MHz, 25°C
Energy Efficiency at peak throughput	3.4Tb/s/W
Peak energy efficiency	7.4Tb/s/W @ 0.6V

Figure 4.18: SSN chip specifications

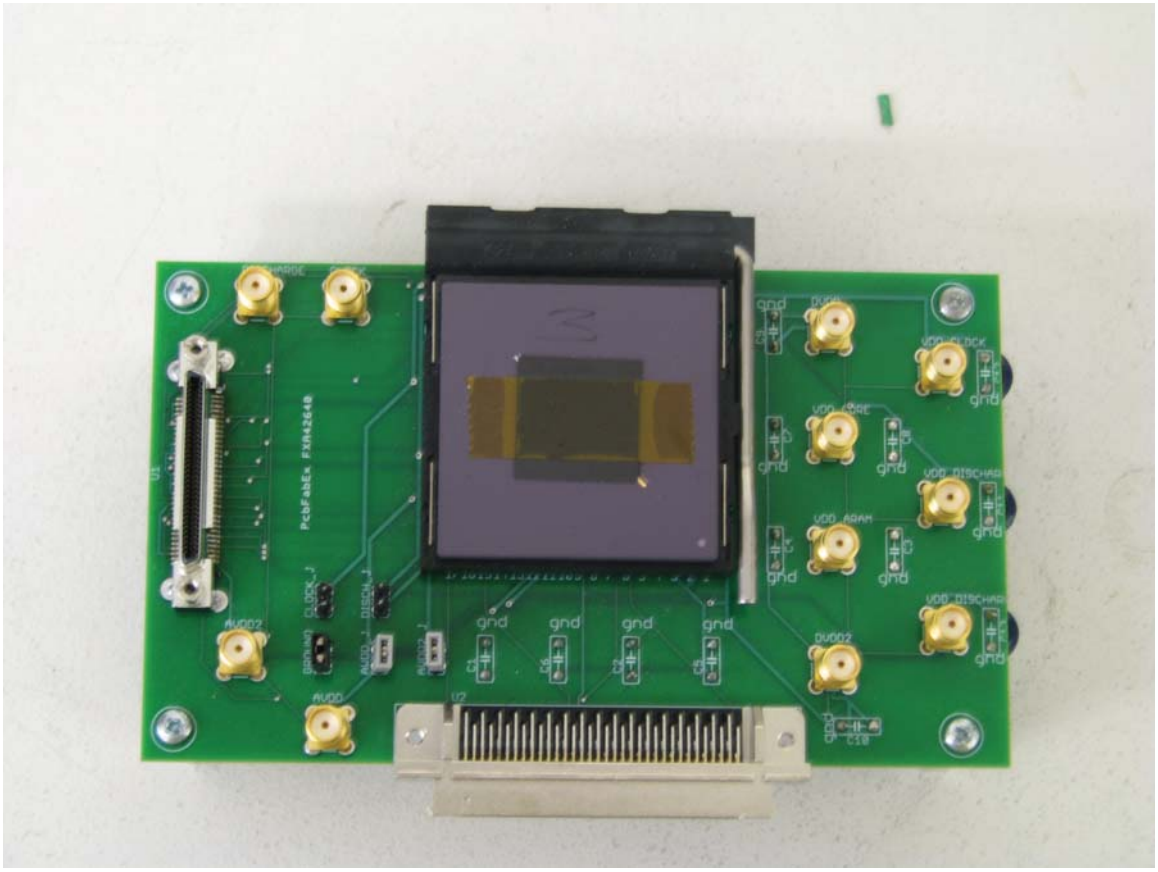

Figure 4.19: PCB hosting SSN test prototype

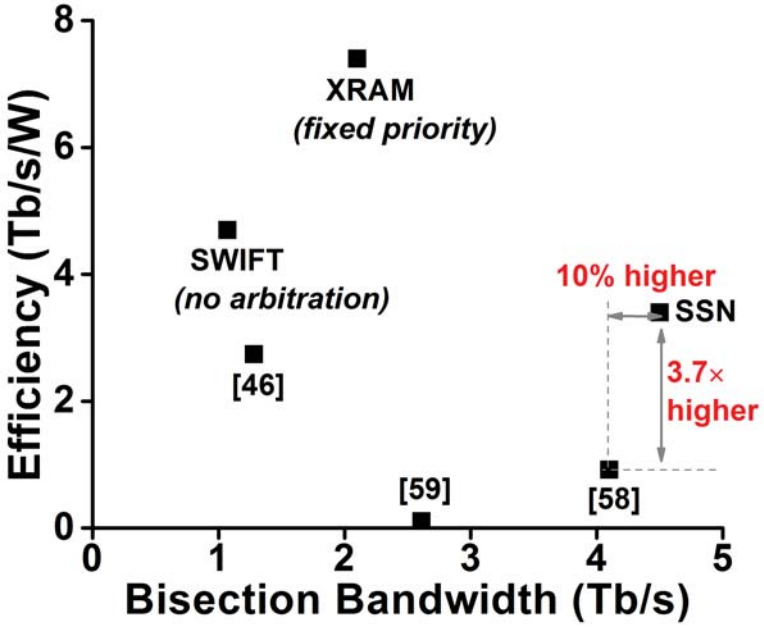

Figure 4.20: Efficiency vs bandwidth plot of recently fabricated high radix switch fabrics

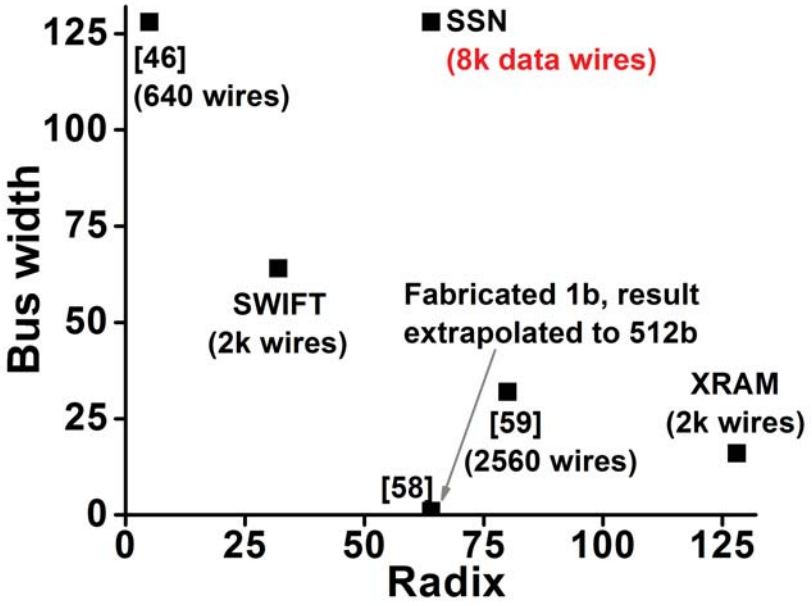

Figure 4.21: Bus width vs radix plot of recently fabricated switch fabrics

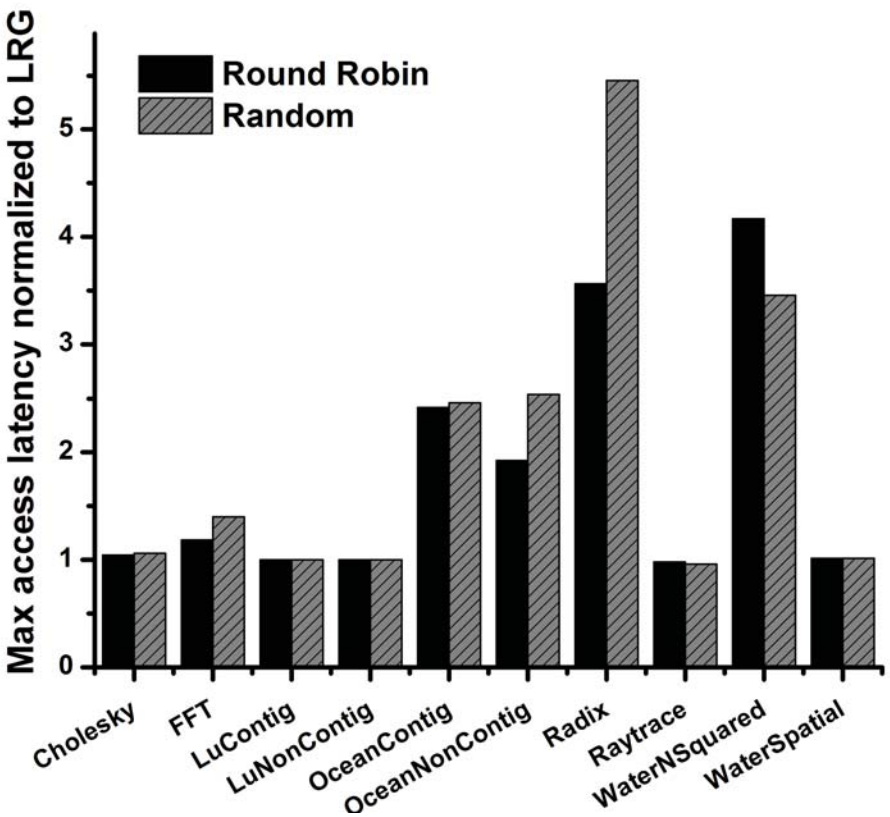

Figure 4.22: Maximum cache access latency of LRG, round robin and random arbitration schemes

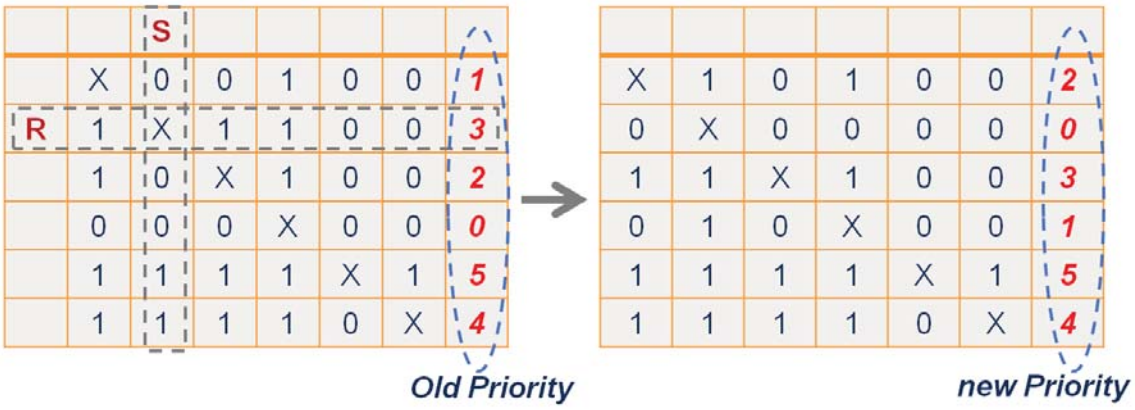

Figure 4.23: LRG based priority update

4.8 Multiple adaptive priority update schemes

4.8.1 Least Recently Granted (LRG)

The priority vectors at various crosspoints along an output bus form a priority matrix. A priority matrix with 6 inputs (arranged top to down numerically from input1 to input6) is shown in Fig. 4.23. Here, the priority line connections are denoted as Xs. The priority matrix satisfies the following criteria: 1) The total number of 0s equals the total number of 1s. 2) Each row has a unique number of 1s which represents the corresponding inputs priority. 3) Each column has a unique number of 1s. 4) At any *priority line* connection (denoted as X in Fig. 4.23), the sum of the number of 1s (or 0s) in its corresponding row and column must add to the total number of inputs.

Though *priority lines* can be randomly assigned to inputs without limiting the generality of the priority update schemes, a diagonal assignment as shown in Fig. 4.23 makes the priority matrix skew (or anti) symmetric and easy to understand. As shown in Fig. 4.23, the input corresponding to second row used the channel most recently. It is assigned a priority level 3. An LRG priority update is accomplished by resetting all priority bits along the second row (denoted by R) and by setting all priority bits (denoted by S) along the second column (which is the priority line for *input2*). In the rest of this section, the other priority update schemes will be explained using the priority matrix notation.

4.8.2 Most recently granted (MRG) or greedy algorithm scheme

For accomplishing an MRG based update, we set all priority bits along the second row (denoted by S) and reset all priority bits (denoted by R) along the second column (which is the *priority line* for the *input2*) as shown in Fig. 4.24. By setting bits in the second row, *input2* now gets the highest priority. Inputs who previously had higher

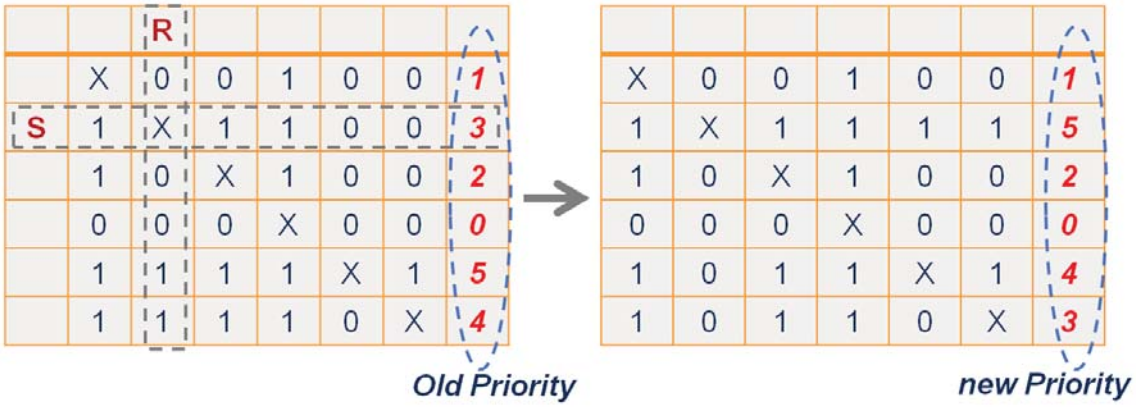

Figure 4.24: MRG based priority update

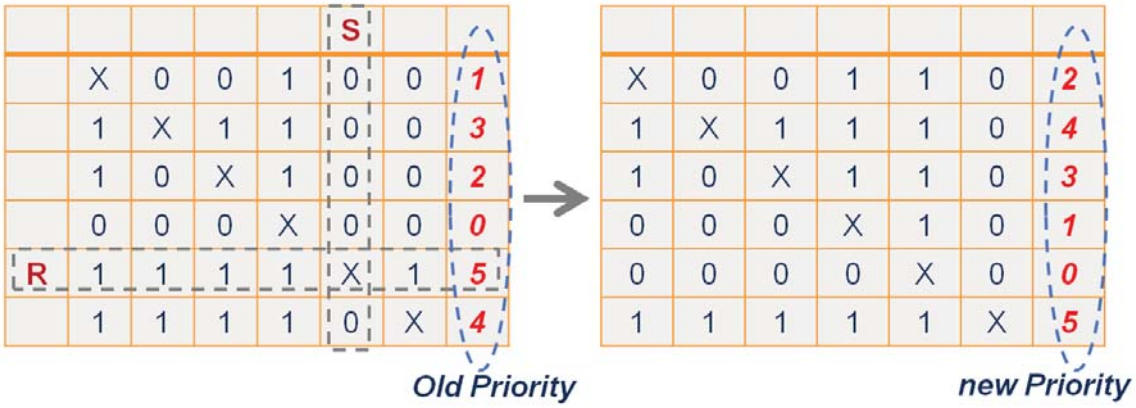

Figure 4.25: Incremental Round Robin based update

priority than *input2* get downgraded by exactly one level. Inputs who previously had lower priority than *input2* retain their old priorities.

4.8.3 Incremental Round Robin

For accomplishing an incremental Round Robin based update, we pick the row with the highest priority. This can be identified by a logical AND operation of all the priority bits. In this case input5 has the highest priority. We reset all priority bits along the fifth row (denoted by R) and set all priority bits (denoted by S) along the fifth column (which is the *priority line* for*input5*) as shown in Fig. 4.25. By resetting

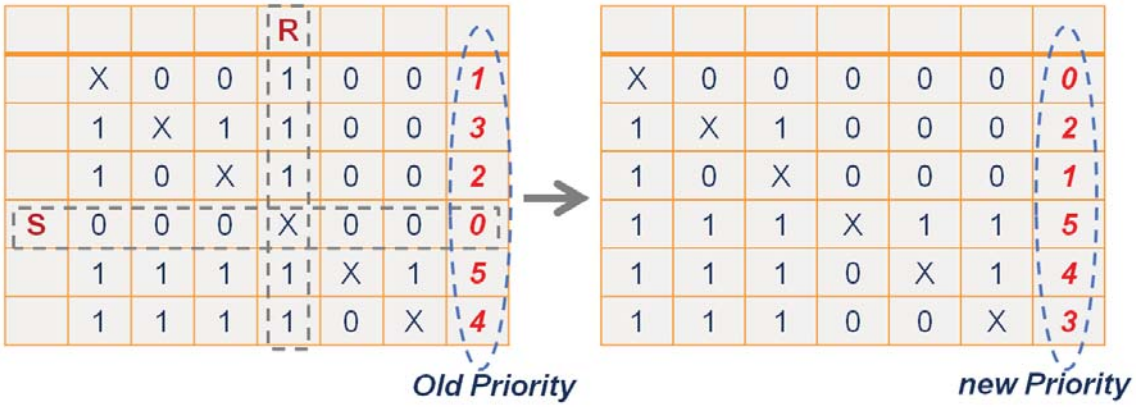

Figure 4.26: Decremental Round Robin based update

bits in the fifth row, *input5* now gets the least priority. All other inputs get upgraded by exactly one level.

4.8.4 Decremental Round Robin

For accomplishing a decremental Round Robin based update, we pick the row with the lowest priority. This can be identified by a logical OR operation of all the priority bits. In this case *input4* has the highest priority. We set all priority bits along the fourth row (denoted by S) and reset all priority bits (denoted by R) along the fourth column (which is the *priority line* for *input4*) as shown in Fig. 4.26. By setting bits in the fourth row, *input4* now gets the highest priority. All other inputs get downgraded by exactly one level

4.8.5 Priority swap

The priorities of 2 inputs can be swapped (without affecting priorities of other inputs) by swapping the priority bits in their corresponding rows and those in the columns corresponding to their *priority lines* as shown in Fig. 4.27. Here, we intend to swap the priorities of *input3* and *input4*. In the physical realization of this technique, already existing word-lines will be used to swap priority bits between columns and

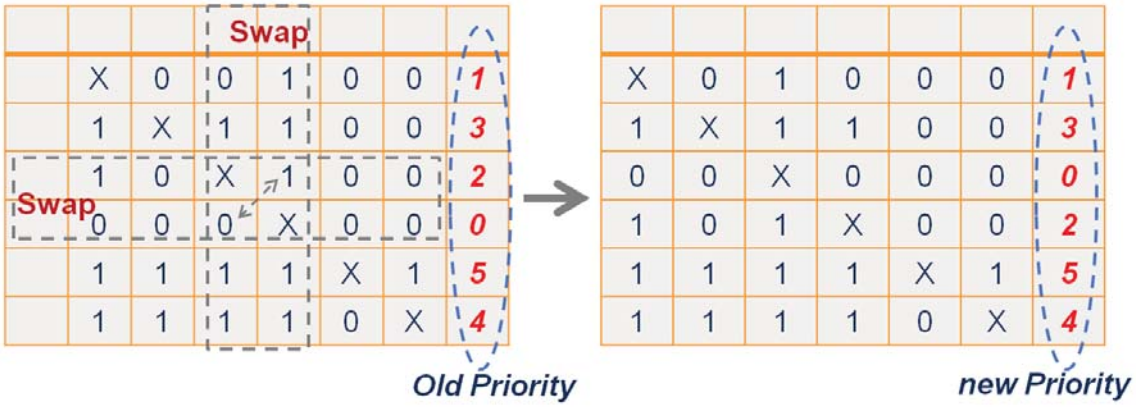

Figure 4.27: Priority swap

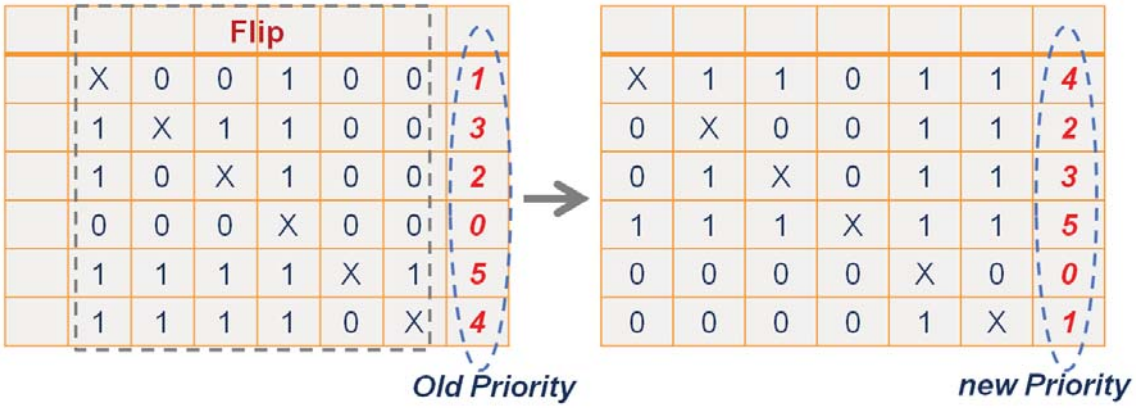

Figure 4.28: Reverse priority order

bit-lines to swap priority bits between rows. In a single cycle, any two priorities can be swapped.

4.8.6 Reverse priority order

The unique priority encoding sheme allows reversing the priority of all inputs instantaneously by flipping all the priority bits as shown in Fig. 4.28. In physical circuit level implementation, rather than flipping all the bits a multiplexer can be used to select the inverted priority. Hence, this functionality can be achieved without the expense of a clock cycle. The consistency of the new priority vectors is guaranteed

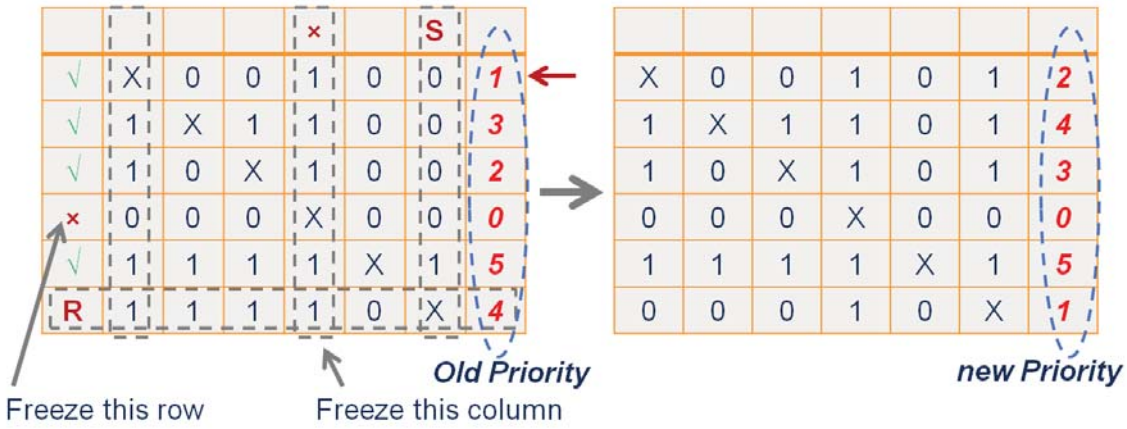

Figure 4.29: Selective least recently granted priority update

because this transformation ensures that the priority matrix still satisfies all the criteria mentioned before.

4.8.7 Selective LRG

In this scheme LRG update is applied to a selective section of inputs. In the standard LRG scheme, the input that used the output bus most recently is downgraded to have the least priority while all inputs with lower priorities get upgraded by exactly one level. In this case *input6* with a priority level 4 used the channel most recently. However, in the selective LRG scheme, instead of downgrading *input6* all the way down to 0, we intend to downgrade it to some intermediate priority (say priority level of *input0* which is 1). To accomplish this, before setting/resetting priority bits we identify certain rows and columns that need to be frozen. In this case, all columns corresponding to priority bits that are high in the first row are frozen as shown in Fig. 4.29. Simultaneously, all rows corresponding to priority bits that are low in the first column (which is the priority line for *input0*) are also frozen. Following this the priority bits in the sixth row are reset (except the bits in frozen columns) and those in the sixth column are set (except the bits in frozen rows). This ensures that the new priority matrix is consistent and the intended priority update is achieved.

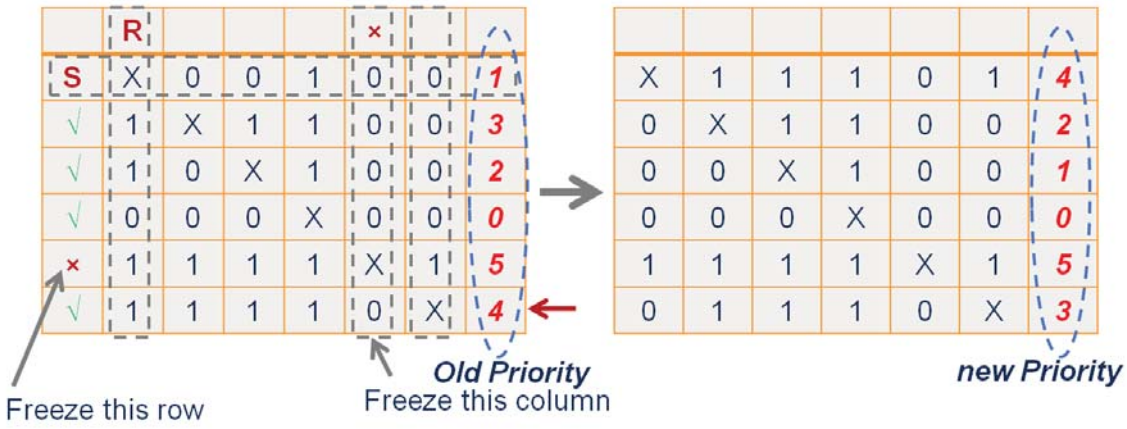

Figure 4.30: Selective most recently granted priority update

4.8.8 Selective MRG

In this scheme MRG update is applied to a selective section of inputs. In the standard MRG scheme, the input that used the output bus most recently is upgraded to have the highest priority while all inputs with higher priorities get downgraded by exactly one level. In this case *input1* with a priority level 1 used the channel most recently. However, in the selective MRG scheme, instead of upgrading input1 all the way up to 5, we intend to upgrade it to some intermediate priority (say priority level of *input6* which is 4). To accomplish this, before setting/resetting priority bits we identify certain rows and columns that need to be frozen. In this case, all columns corresponding to priority bits that are low in the sixth row are frozen as shown in Fig. 4.30. Simultaneously, all rows corresponding to priority bits that are high in the sixth column (which is the priority line for *input6*) are also frozen. Following this the priority bits in the first row are set (except the bits in frozen columns) and those in the first column are reset (except the bits in frozen rows). This ensures that the new priority matrix is consistent and the intended priority update is achieved.

4.9 Summary

In this chapter, we presented SSN that leverages a novel priority encoding scheme that re-uses existing logic and interconnect resources in switch fabric to locally store priorities at router crosspoints resulting in a compact implementation. A 64×64 SSN with 128b data bus achieves a peak throughput 4.5Tb/s at an energy efficiency of 3.4Tb/s/W while spanning only 4.06mm^2 in 45nm SOI CMOS. It features a single cycle least recently granted arbitration technique that re-uses data buses and switching logic, a 4-level message based priority arbitration for quality of service and unique bi-directional bit-line repeaters to aid scalability. The unique priority encoding scheme also allows seamless implementation of many other arbitration policies in addition to LRG with very minimal overhead.

CHAPTER V

TABS : Thyristor Assisted Bi-directional Signalling

5.1 Introduction

The switch fabrics presented in previous chapters enable single stage high radix implementation of system level interconnect network. Such a network provides uniform latency making it easy to improve application runtime and guarantee quality of service. However, the average routes to and from different IPs through the switch fabric increases. This results in more energy being spent while communicating to and from the switch fabric. In this chapter, we address this issue by proposing a novel repeater called TABS (thyristor assisted bi-directional signaling) as shown in Fig. 5.1. TABS is a thyristor-assisted standard cell compatible self-timed bi-directional repeater with no configuration overhead. It enables 8mm interconnects to achieve 37% higher speed at 20% lower energy over conventional repeaters in 65nm CMOS at 1.0V. In TABS, absence of configuration logic removes the need for clocking, yielding up to 14× higher energy efficiency at very low data switching activity.

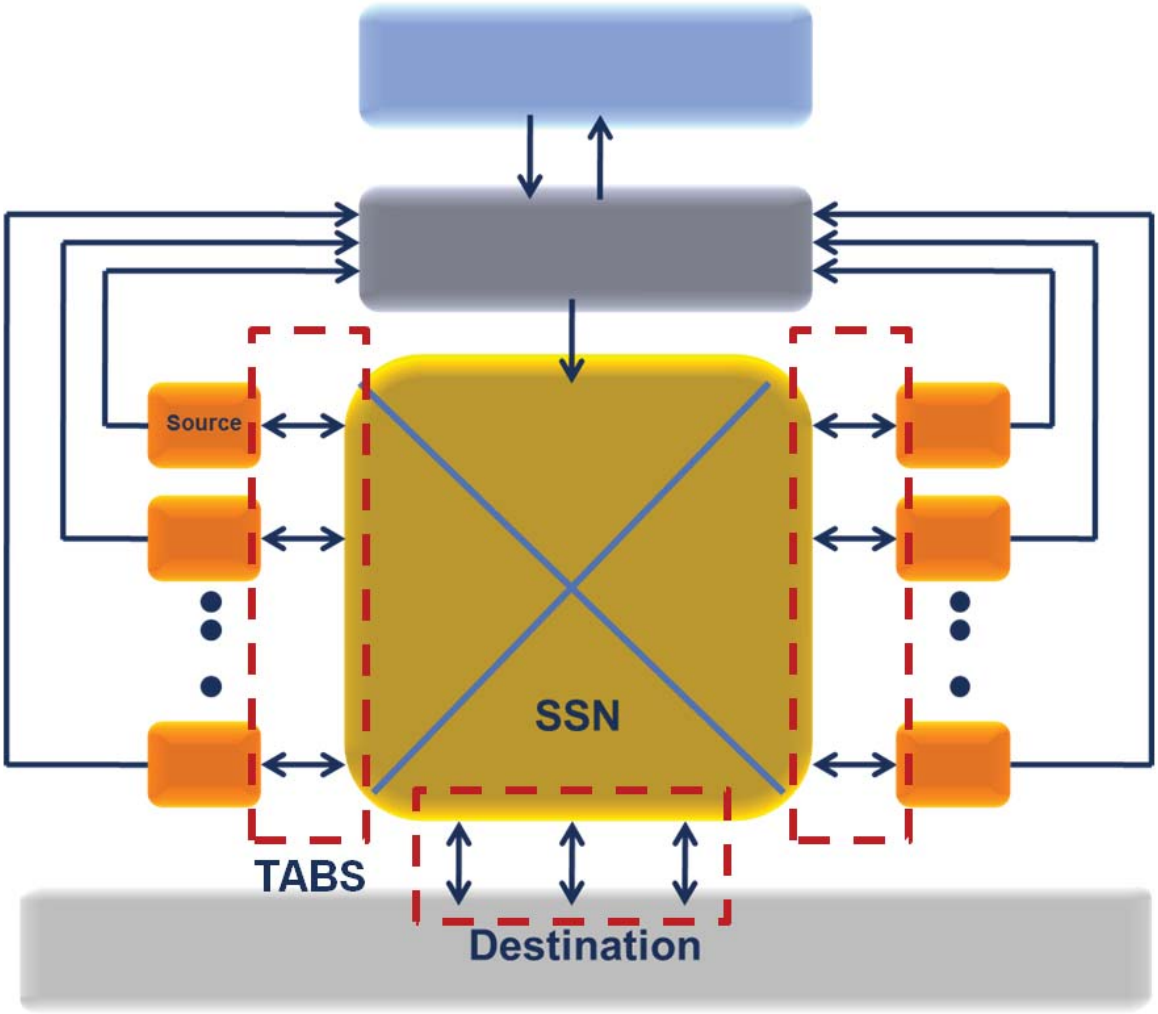

Figure 5.1: TABS: Thyristor assisted bi-directional signalling

5.2 Motivation

Bi-directional interconnects are an integral part of global communication networks in multiprocessor chips. They facilitate high bandwidth with low silicon overhead by eliminating the need for replicating unidirectional signal wires [51] [61]. Conventional bi-directional links are based on duplication of unidirectional repeaters, one of which is selectively activated for signal propagation. This implementation incurs logic and interconnect overhead to configure the repeaters, degrading performance and energy efficiency. Additionally, a synchronizing signal in the form of a clock is needed to eliminate contention when reversing signal propagation direction. Further, a bi-directional link can be driven at multiple locations in many snoop-based signaling schemes, making signal propagation information challenging to obtain *a priori* [62]. Recent custom-designed repeater-less signaling techniques have achieved high speed with low energy dissipation based on reduced voltage swing [63] [64] [65] [66]. However, such techniques typically require careful custom design that is tailored to each specific interconnect situation and involves precise device matching, additional supply voltages, and wider wire thickness/pitch. Hence they cannot be easily used in synthesis-based design flows or re-used in different interconnect situations in the same design.

5.3 TABS approach

Instead, we target a drop-in replacement for conventional repeaters within a standard cell based design flow. To this end, we present a thyristor-assisted bi-directional signaling (TABS) technique with the following key features. 1) The fully static circuit implementation allows TABS to be used as a standard cell in synthesis-based design flows; 2) The self-timed bi-directional repeater has no configuration overhead and enables 8mm interconnects to achieve 37% higher performance at 20% less energy

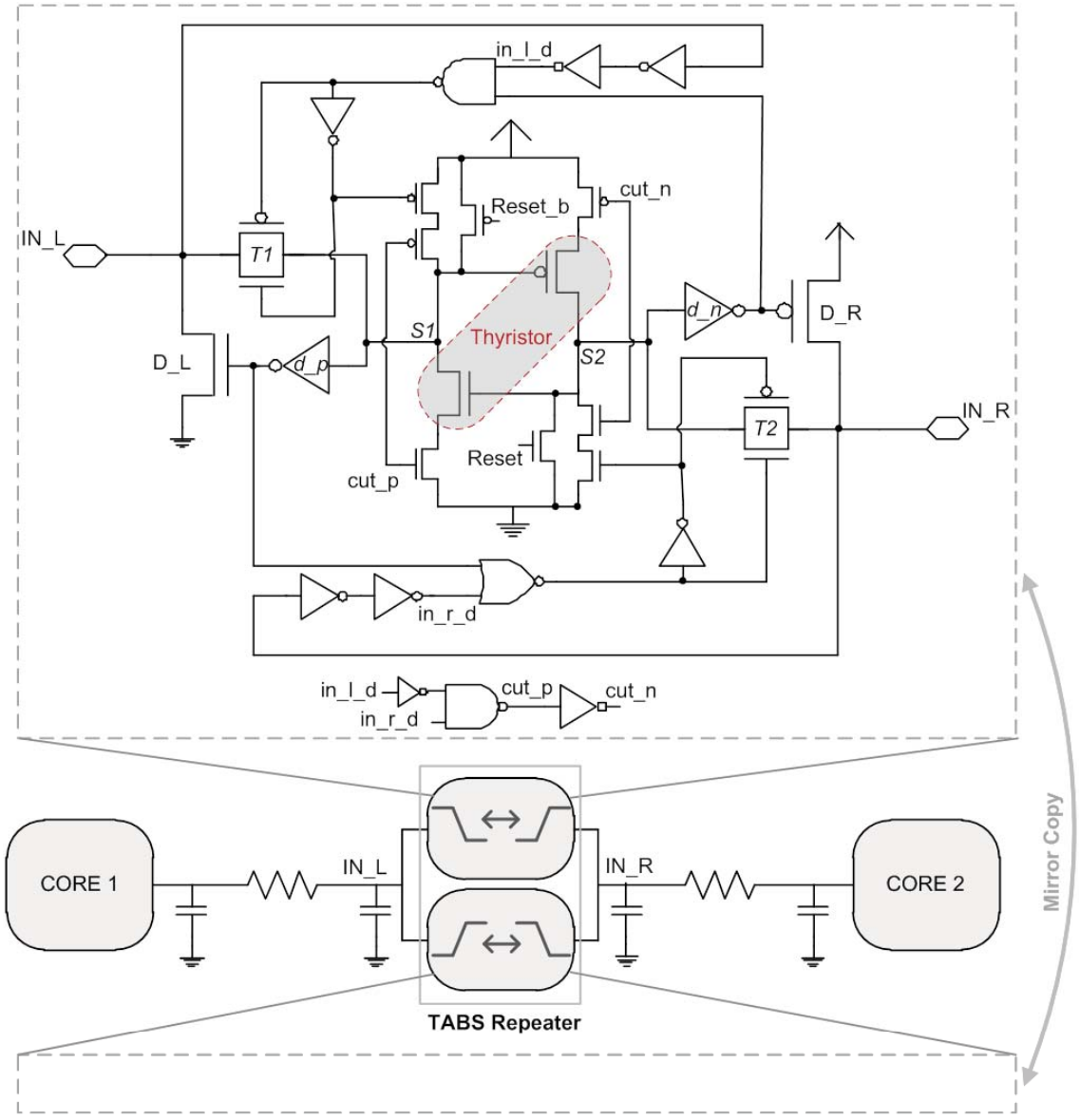

Figure 5.2: TABS repeater with thyristor sensing element and no additional configuration logic

per bit compared to conventional repeaters in 65nm CMOS at 1.0V; 3) Absence of configuration logic obviates the need for clocking, simplifying the design flow and providing up to 14× better energy efficiency for low data switching activity; 4) Robust operation across 0.6 to 1.2V supply. Besides improving performance, these features increase the optimal repeater insertion interval for TABS from 625μm to 1mm, yielding 38% fewer repeaters in the signal propagation path. At 1.5mm spacing TABS energy efficiency is improved by 51% and repeater count reduced by 58% while maintaining iso-performance with conventional optimally spaced repeaters. We also present a synchronous version of the repeater for use at synchronizing boundaries in place of conventional bi-directional flip-flops, saving 36% area.

Fig. 5.2 shows a TABS half-repeater with a pair of NMOS and PMOS transistors arranged in a CMOS thyristor configuration and used as a transition amplifier. The expanded circuit senses a falling transition on *IN_L* and/or a rising transition on *IN_R*, thereby causing the thyristor to switch and pull both interconnect nodes to opposite supply rails. Two such repeaters are used in parallel with their ports connected to opposite nodes of the link to form a TABS bi-directional repeater.

5.4 TABS operation

Each TABS repeater has 3 operating states: *High gain*, *low gain*, and *passive*. In the absence of any switching or when *reset* is asserted, the repeater is in *low gain* state with the thyristor nodes pre-charged/pre-discharged to voltage levels where the sensing transistors have their lowest gain (Vgs = 0). This is shown in Fig. 5.3. The sensing nodes *(S1/S2)* are connected to *IN_L* and *IN_R* through transmission gates *(T1/T2)* that are turned ON. *S1/S2* node voltages are held by keepers placed on *IN_L* and *IN_R* (not shown). When *IN_L* transitions from high to low, node *S1* initially follows it. The PMOS device in the thyristor gradually turns ON, raising *S2* which causes the NMOS thyristor device to turn ON. This regeneration mechanism causes *S1*

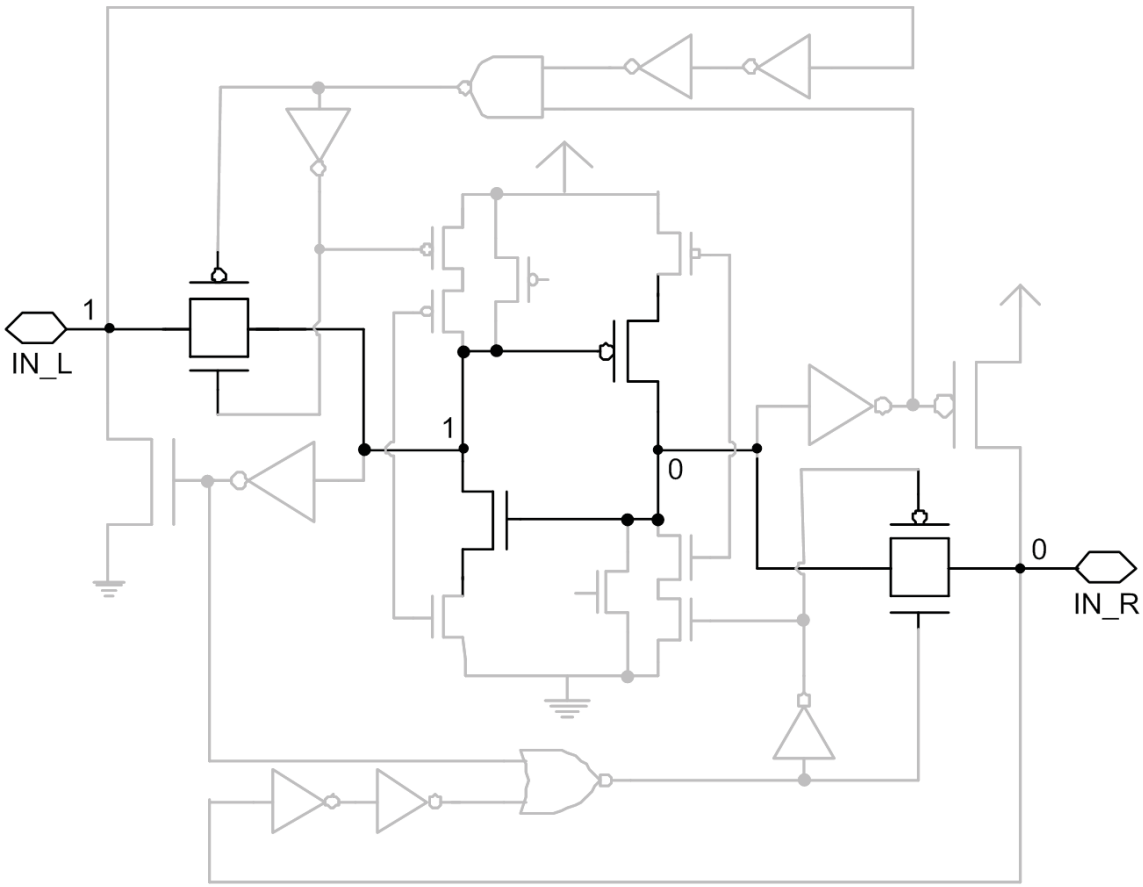

Figure 5.3: TABS low gain state

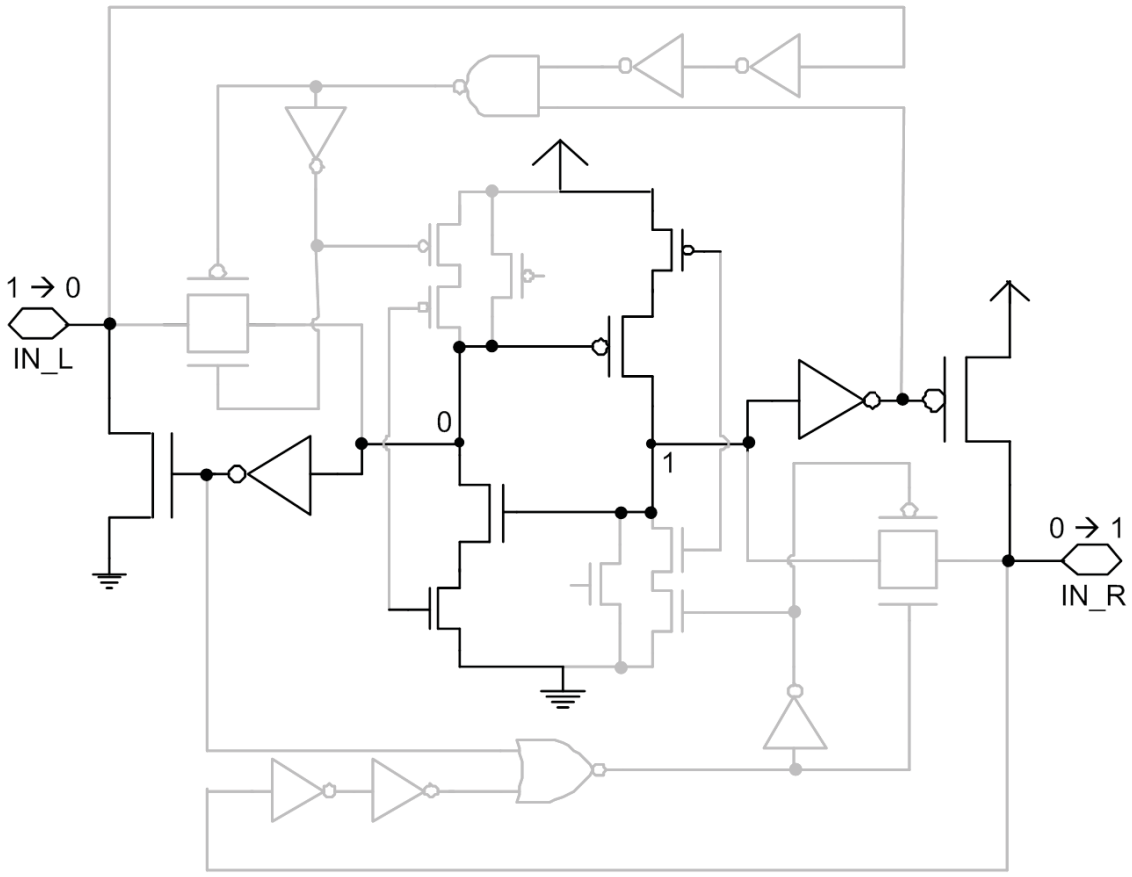

Figure 5.4: TABS high gain state

and *S2* to switch rapidly. Inverter *d_n (d_p)* is skewed with a stronger NMOS (PMOS) to provide a faster response to *S1/S2* switching and quickly turn OFF transmission gates *T1 (T2)*. This decouples *S1 (S2)* from *IN_L (IN_R)* and allows for faster switching. Once the transistors in the thyristor transition, the repeater enters the *high gain* state where the sensing transistors have highest gain (Vgs = VDD/VDD), causing large internal drivers *(D_L/D_R)* to rapidly pull *IN_L* and *IN_R* towards supply rails. This is shown in Fig. 5.4. Once *IN_L* and *IN_R* have both transitioned, their delayed signals *in_l_d* and *in_r_d* enable the pre-charge signals *cut_n* and *cut_p* which reset and hold the thyristor in its precharge state. The repeater is now in *passive state* with its thyristor disconnected from *IN_L* and *IN_R*. Finally, when *IN_L* and *IN_R* transition back, the repeater again enters *low gain state*.

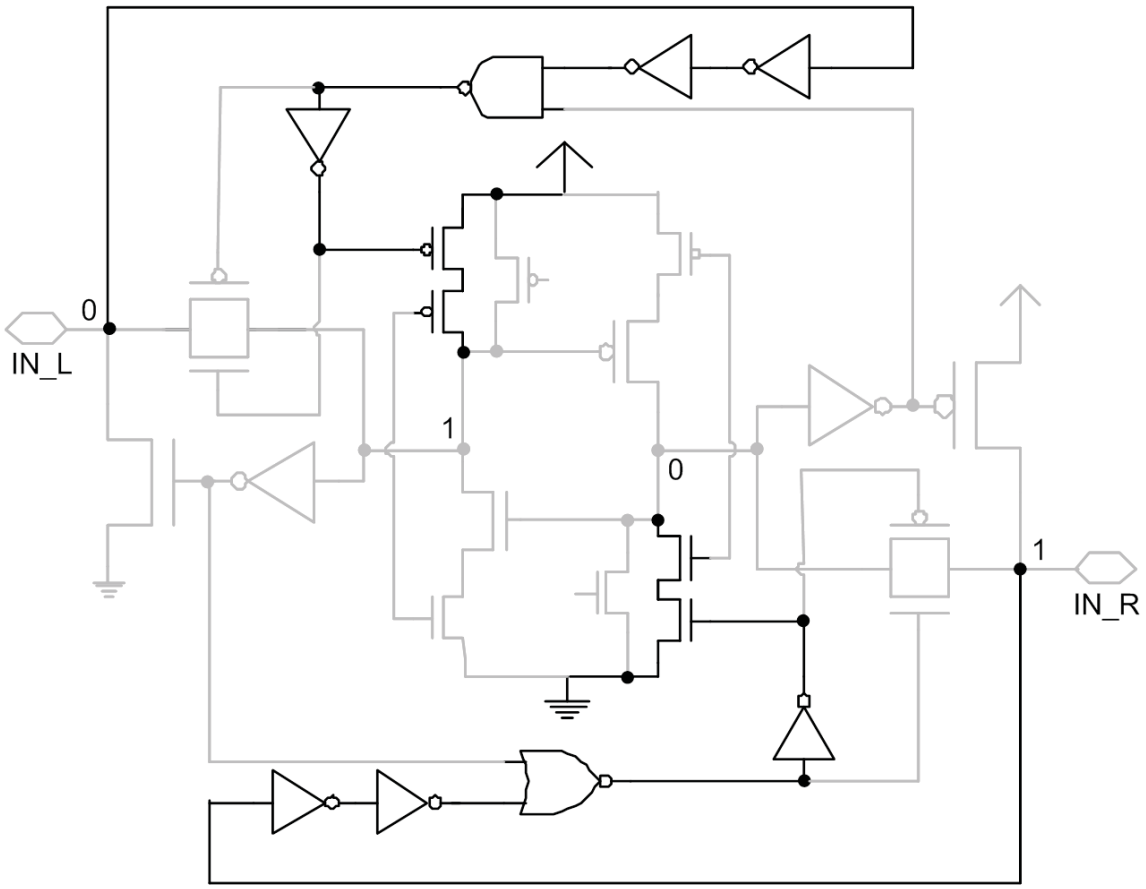

Figure 5.5: TABS passive state

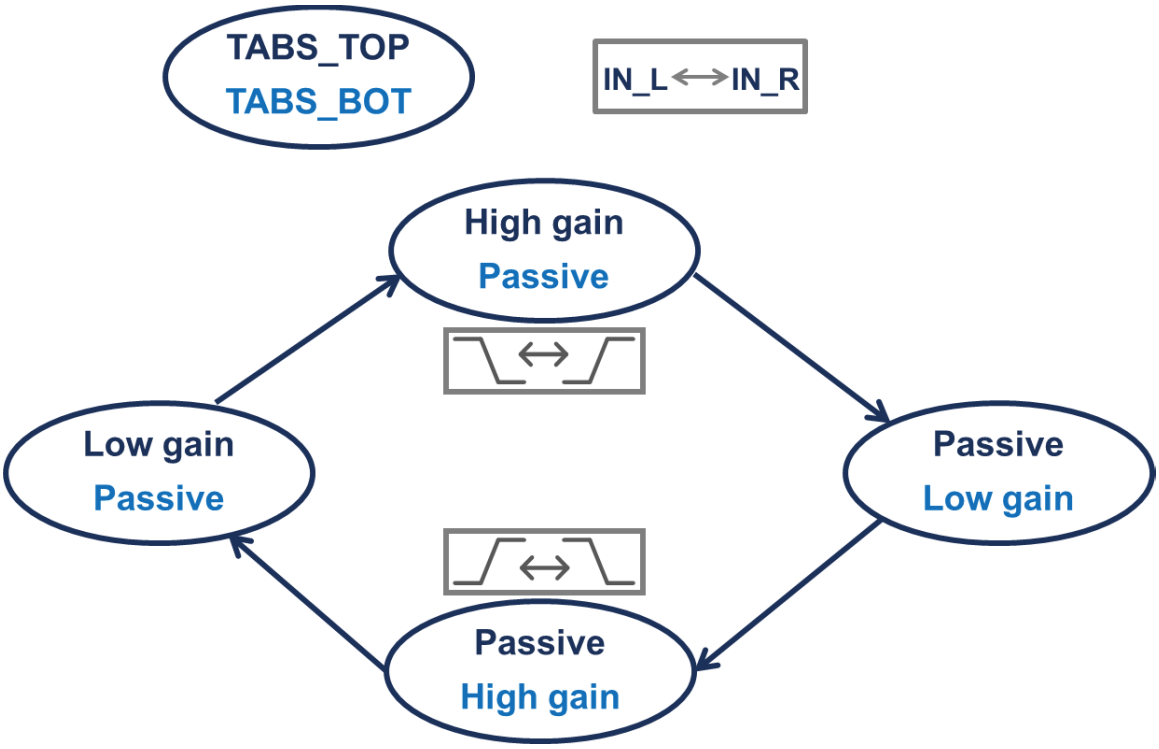

Figure 5.6: TABS state transition diagram

While one repeater is in the *low gain* state, the other half-repeater is in the *passive* state. A transition on the interconnect causes the sensing repeater to switch from *low gain* to *passive* state via the *high gain* state while the other repeater transitions from *passive* to *low gain* and begins sensing the interconnect to detect the next transition. This is shown in Fig. 5.5. The state transition of both repeaters is shown in Fig. 5.6.

Using the same circuit technique, a bi-directional latch is also designed for data hand-off at synchronizing boundaries as shown in Fig. 5.7. The direction is configured into a 6T SRAM based on the state of the interconnect at the falling edge of clock, which is later used in the positive phase for sensing the appropriate transition.

TABS offers 3 key advantages over inverter-based repeaters: 1) The regeneration followed by decoupling mechanism makes switching latency less dependent on slow slew rates on long interconnect as commonly seen in global wires; 2) In a conventional repeater a transition is performed only by the driver, as opposed to TABS where the

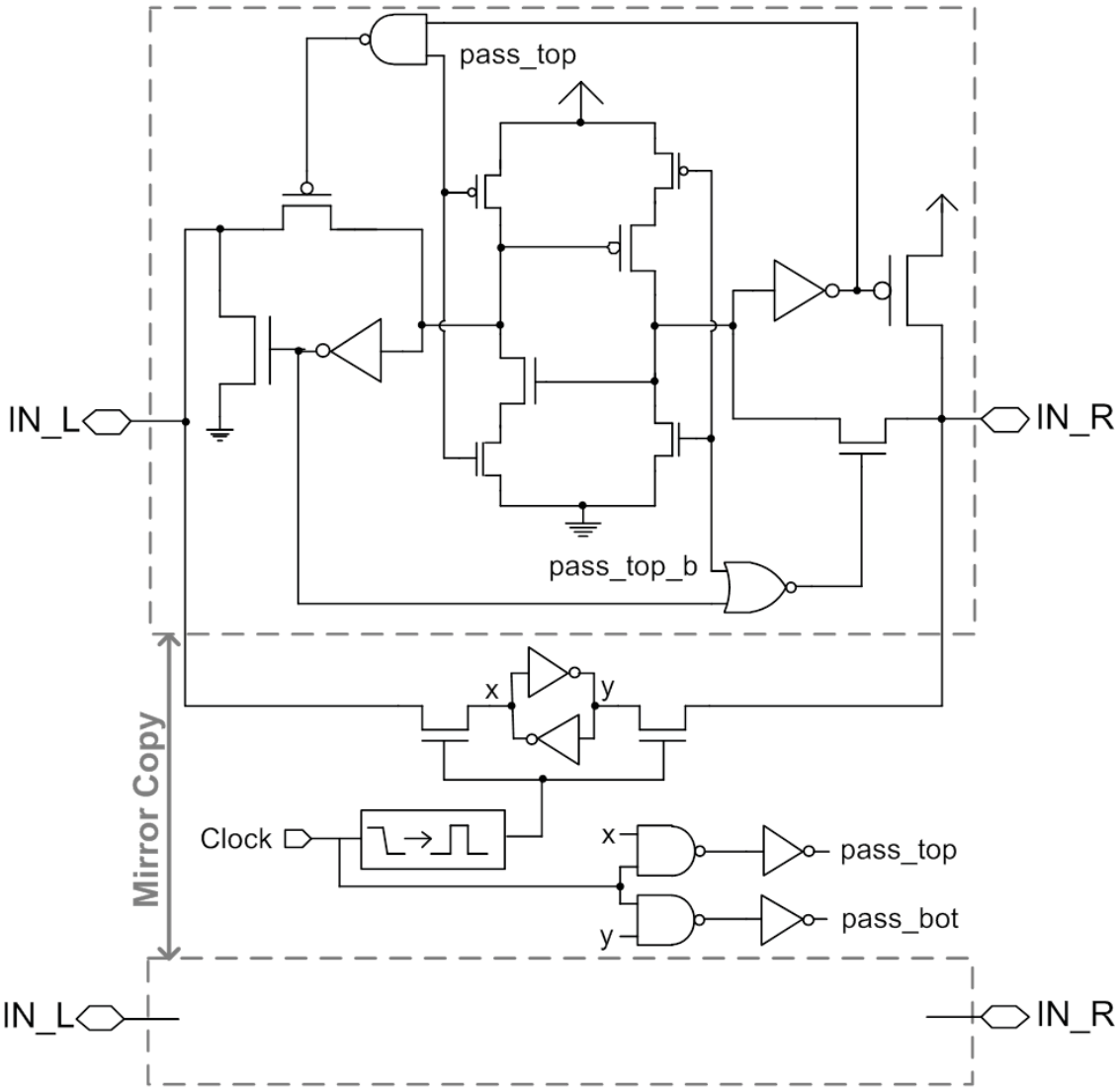

Figure 5.7: TABS based bi-directional latch

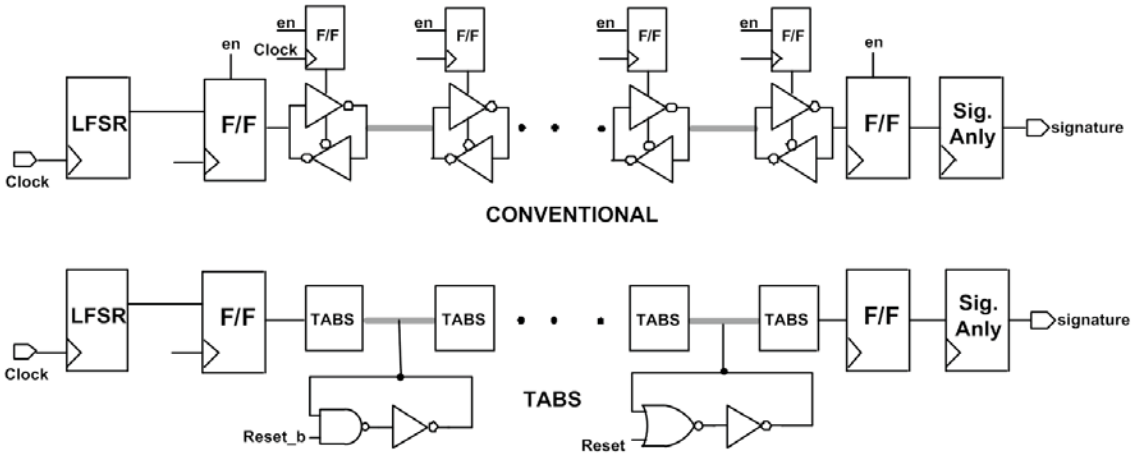

Figure 5.8: TABS and conventional test structures

receiver aids the driver after detection; 3) The internal feedback and state tracking mechanism eliminates the need for a global synchronizing signal.

5.5 Test prototype

To compare TABS repeater with conventional standard cell based repeaters, we fabricated a test prototype with varying links driven with both repeater topologies in 65nm bulk CMOS. Fig. 5.8 shows the schematic for both structures. In both links data can be driven in either directions. For conventional repeaters, the direction is locally stored in each repeater in a flip flop, thereby necessitating the need for clock tree. In TABS repeater, the configuration logic is embedded within thereby obviating the need for clock. Fig. 5.9 shows the prototype's die photograph and Fig. 5.10 shows the printed circuit board hosting TABS test prototype.

5.6 Results

Fig. 5.11 shows simulated power/delay curves for an 8mm link with varying repeater sizes. Optimal insertion interval is 1000μm for TABS and 625μm for conven-

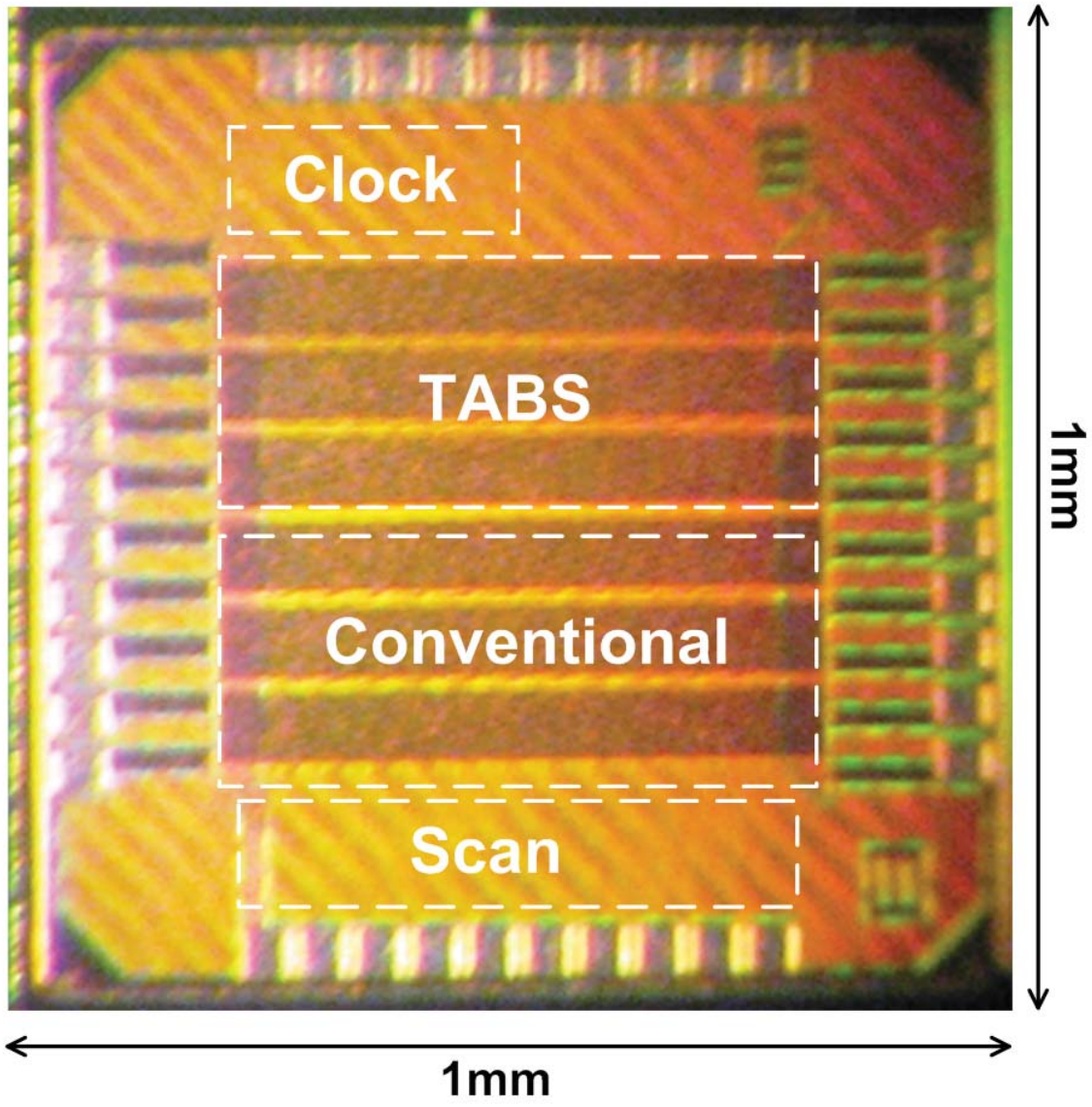

Figure 5.9: TABS prototype die photo

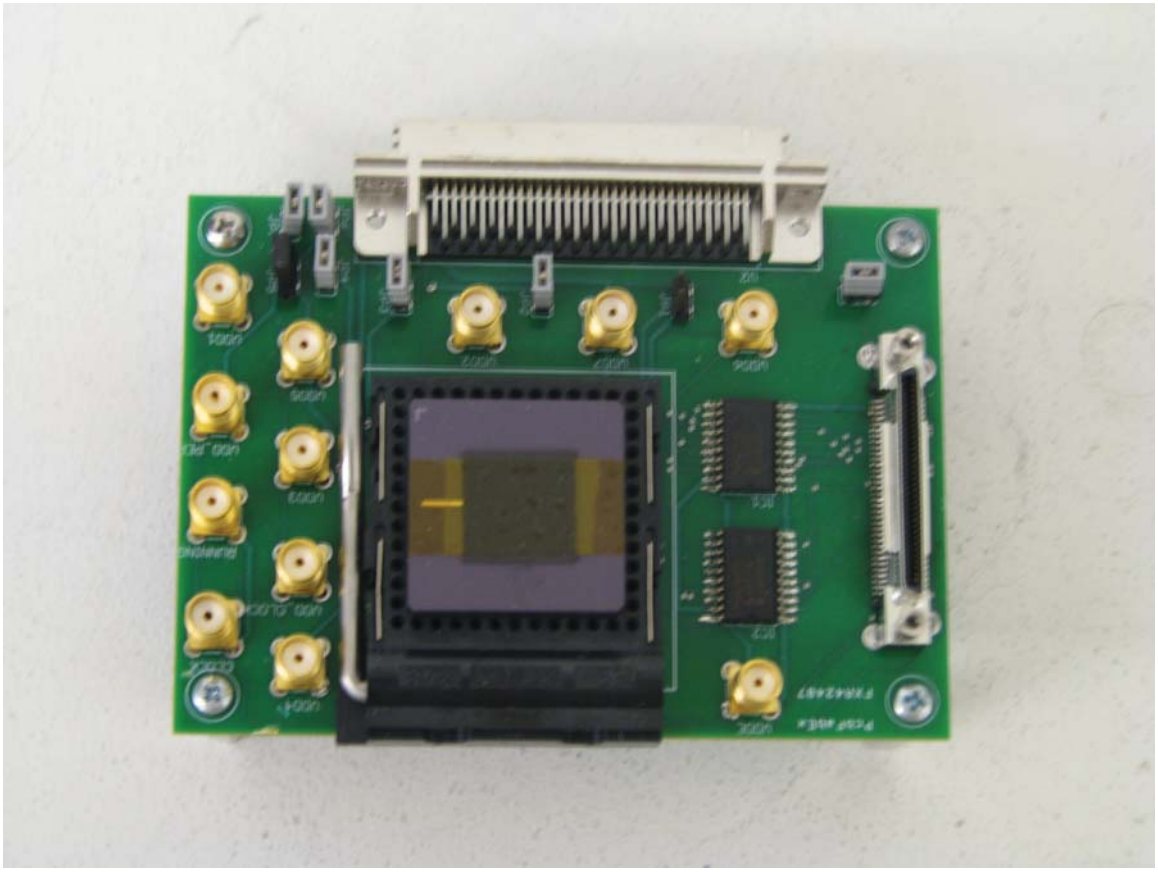

Figure 5.10: PCB hosting TABS prototype

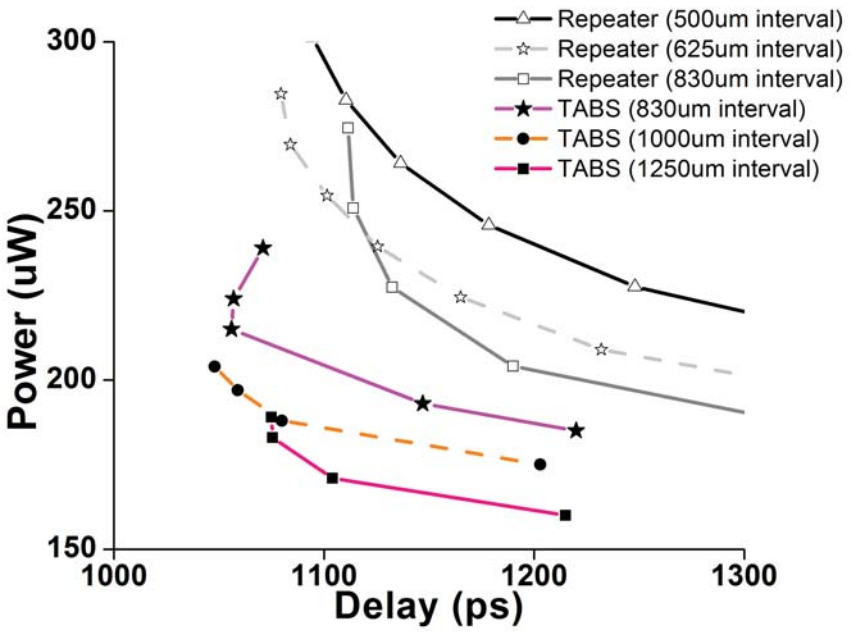

Figure 5.11: Simulated power delay curves for TABS and conventional repeater

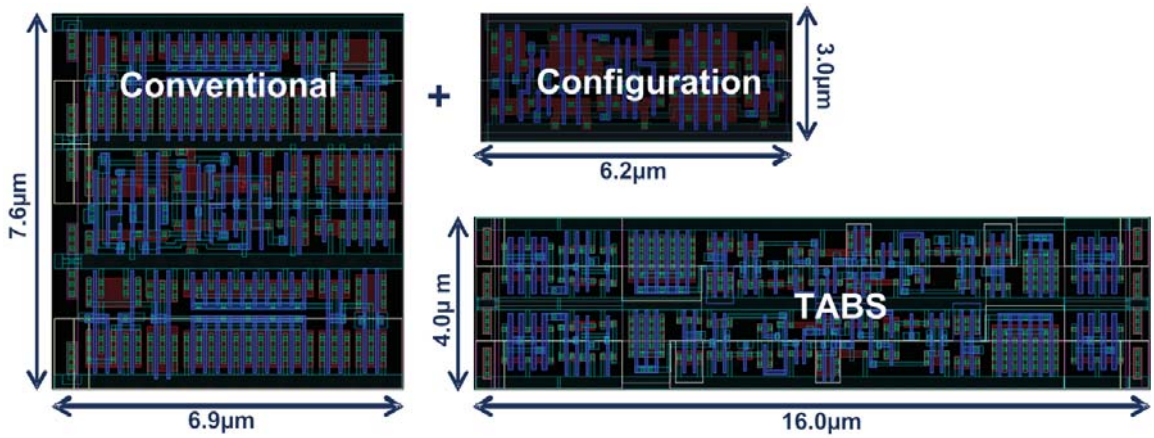

Figure 5.12: TABS and conventional repeater layouts

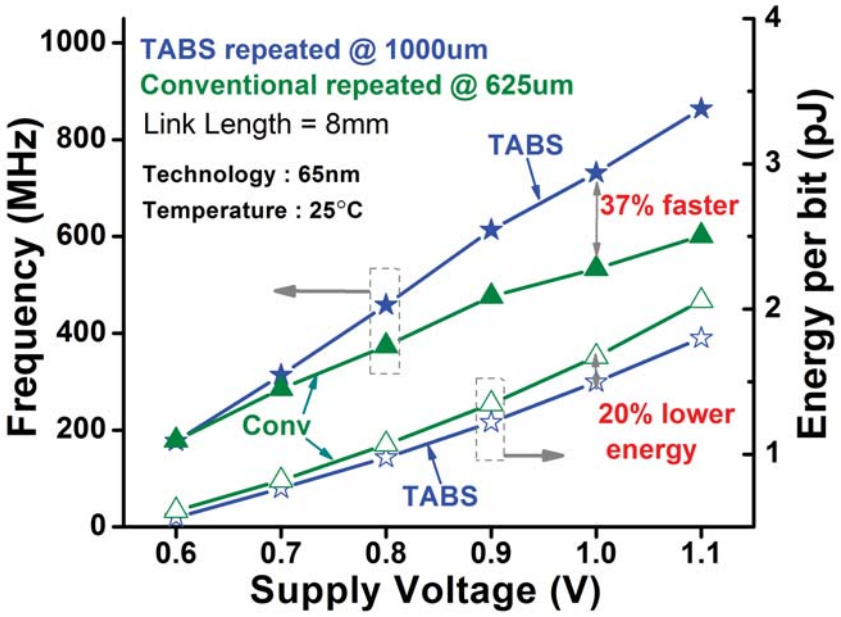

Figure 5.13: Measured performance and energy for 8mm long interconnect with TABS (1mm insertion interval) and conventional repeater (625μm insertion interval) at 50% bi-directional traffic.

tional repeaters. As shown in Fig. 5.12, TABS spans 11% less area than conventional repeaters when configuration overhead is taken into consideration.

A test prototype was fabricated in 65nm bulk CMOS where TABS was treated as a standard cell replacement for traditional repeater/buffers in the design flow. Both links were implemented using M4 and M5 with 100nm wire width (the minumum in 65nm) and 200nm pitch (the minimum in 65nm) and worst-case aggressor switching. The TABS-enabled link operates at 732MHz, which is 37% faster than the conventional link at 1.0V while dissipating 20% less energy as shown in Fig. 5.13. Due to the absence of any configuration overhead TABS improves energy efficiency by 27% to 47% with increasing bi-directional traffic at iso-throughput of 602Mb/s per link as shown in Fig. 5.14.

The absence of clock facilitates seamless signaling across different frequency domains and improves TABS energy efficiency by up to 14× at a low, 0.005 data switch-

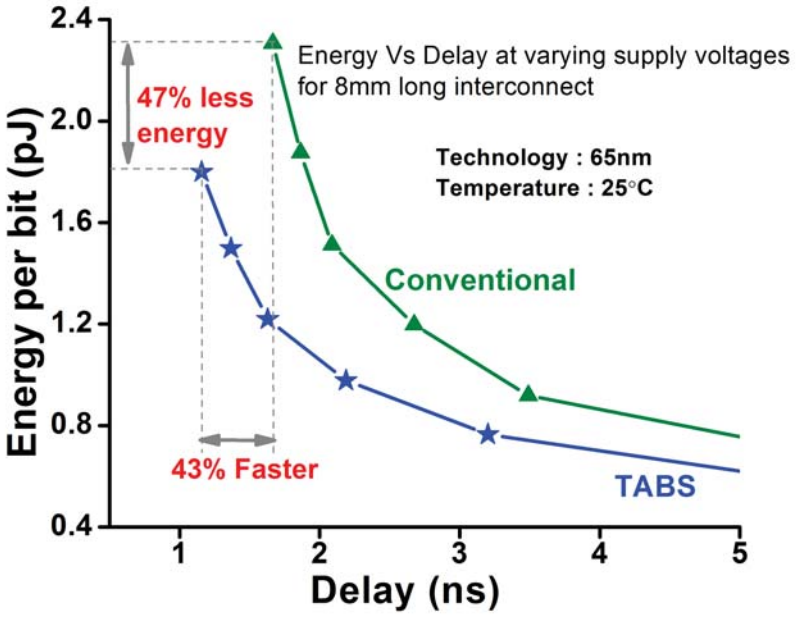

Figure 5.14: Measured energy versus delay curve for TABS and conventional repeaters. TABS is 43% faster and dissipates 47% lower energy at 1.1V and 100% bi-directional traffic

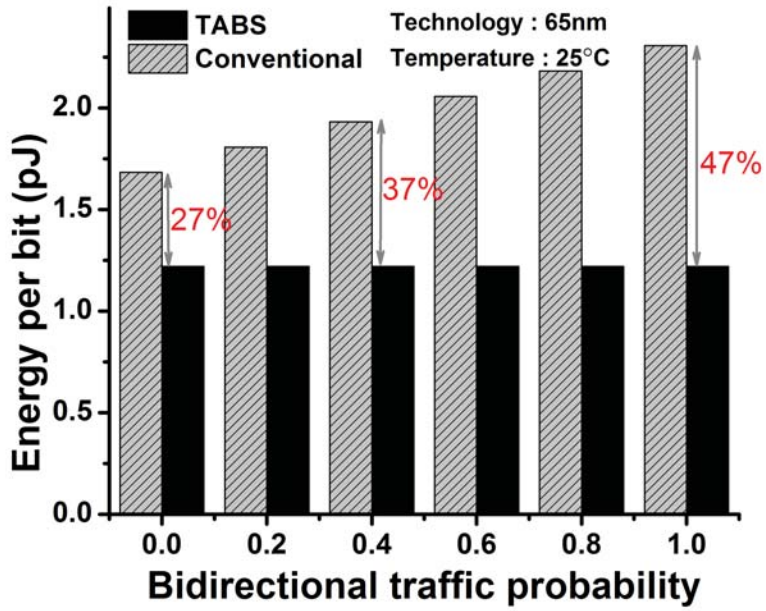

Figure 5.15: Measured energy dissipation in TABS and conventional repeaters with varying percentage of bi-directional traffic at iso-performance at 1.1V.

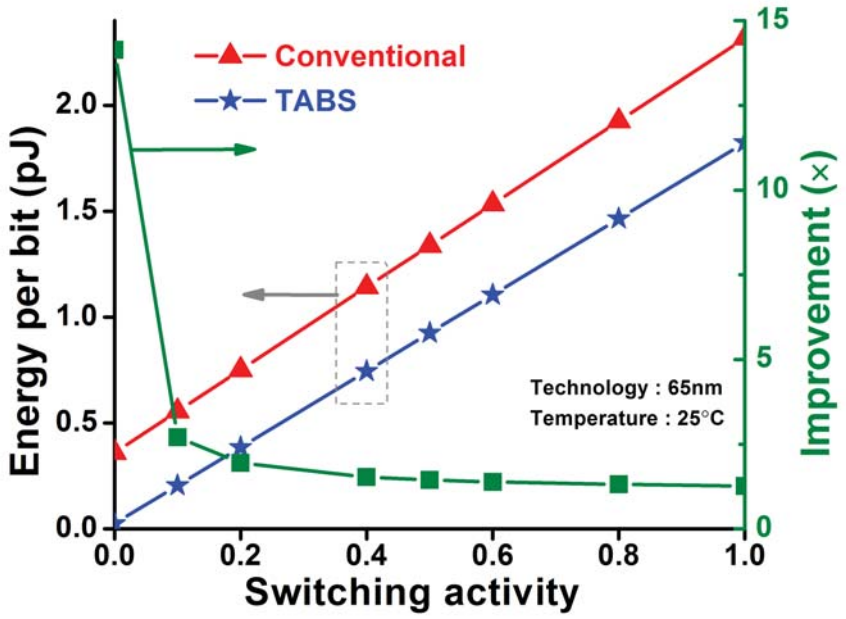

Figure 5.16: Measured energy with varying switching activities at 1.1V. TABS energy savings improve from 27% to 14× as switching activity reduces.

ing activity over conventionally clocked repeaters as shown in Fig. 5.15 and Fig. 5.16. TABS improved performance can be exploited for further energy efficiency gains by increasing repeater insertion interval. Fig. 5.17 shows measured energy vs. delay curves for TABS at 1.5mm insertion interval along with conventional repeaters with varying bi-directional traffic rate. At 1.5mm insertion interval TABS energy efficiency further improves up to 51% over conventional repeaters at iso-performance while using 58% fewer repeaters. This makes TABS highly suitable for links that run over caches and other IP blocks that prohibit frequent repeater insertion. TABS is fully functional across a temperature range of -20^{0}C to 90^{0}C as shown in Fig. 5.18 at 1.0V.

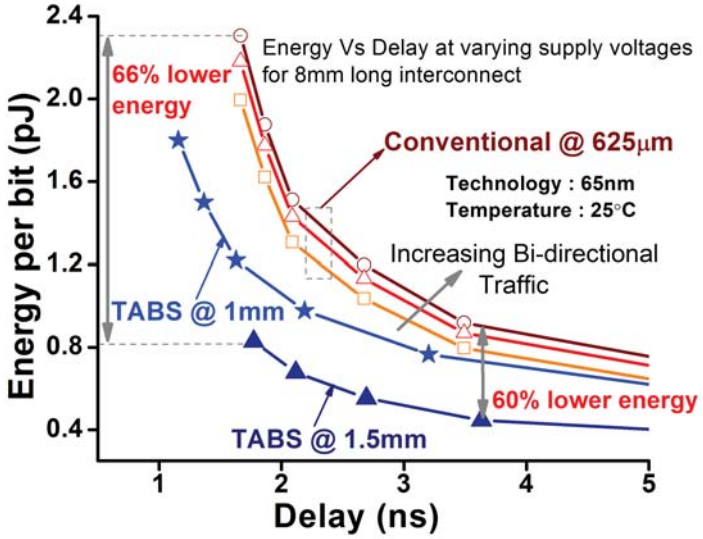

Figure 5.17: Measured energy vs. delay curves for TABS inserted every 1mm and 1.5mm, and conventional repeaters optimally inserted every 625μm. Energy efficiency improves by 51% over conventional link at iso-performance at 1.0V with 58% fewer repeaters by increasing TABS insertion interval to 1.5mm

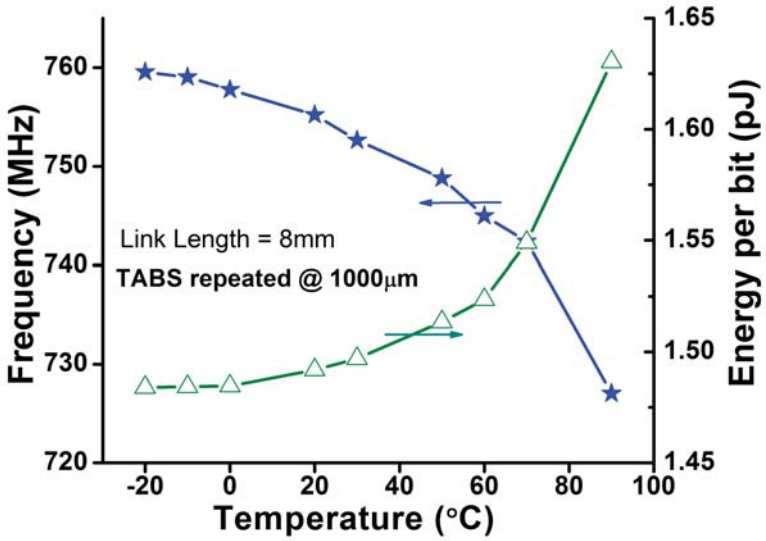

Figure 5.18: Measured performance and energy for TABS at different temperatures at 1.0V.

CHAPTER VI

Conclusion and Future Work

6.1 Summary

With core count scaling up in modern day multiprocessor systems, on-die interconnect fabrics have become one of the limiting factors in improving computing efficiency. This challenge is compounded by the fact that unlike other modules the complexity of switch fabrics grows quadratically with their size. In this dissertation, we proposed many circuit and architectural techniques to improve fabric scalability in the face of this challenge. In chapter 2, we introduced a new permutation network called XRAM to optimize data routing across the fabric. XRAM uses an SRAM-based approach that results in a compact silicon footprint that scales well with network dimensions. It supports all permutations and multicasts. Capable of storing multiple shuffle configurations and aided by a novel sense-amp for robust bit-line evaluation, a 128×128 XRAM with 16b data bus fabricated in 65nm CMOS achieves a bandwidth exceeding 1Tbit/s, enabling a 64-lane SIMD engine operating at 0.72V to save 46.8% energy over an iso-throughput conventional 16-lane implementation operating at 1.1V.

XRAM is very efficient in handling deterministic traffic flow which is a key feature of digital signal processing applications. However, for random traffic arbitration becomes the bottleneck in XRAM. Hence, in chapter 3 we introduced SWIFT that uses circuit techniques to integrate the arbiter within the permutation network. SWIFT

retains all XRAM data permutation capabilities in addition to its ability to perform high radix arbitration without incurring additional delay. It co-optimizes arbiter and crossbar logic using a unique fabric architecture that integrates conflict resolution with data routing to optimally use logic and interconnect resources.

SWIFT uses a fixed set of priority vectors for detecting and resolving conflicts during arbitration. Although chances are slim, it is prone to starvation and throughput degradation under pathogenic traffic patterns. We address this issue in chapter 4 by introducing SSN, which is a fabric architecture to accomplish the least recently granted arbitration scheme by reusing already existing logic/interconnect resources in the fabric. Building on that, we also propose novel schemes to accomplish a variety of arbitration policies with very minimal overhead.

With cores getting simpler and fabric complexity getting mitigated using the above mentioned techniques, communication to and from the interconnect fabric becomes the next limiting factor. A high radix fabric enables high degree of IP integration. However, the average routes to and from different IPs through the switch fabric increases. This results in more energy being spent while communicating to and from the switch fabric. To address this, in chapter 5 we introduced a novel repeater called TABS (thyristor assisted bi-directional signaling), which is a standard cell compatible self-timed bi-directional repeater with no configuration overhead. It enables 8mm interconnects to achieve 37% higher speed at 20% lower energy over conventional repeaters in 65nm CMOS at 1.0V. In TABS, absence of configuration logic removes the need for clocking, yielding up to 14× higher energy efficiency at very low data switching activity.

6.2 Future research directions

The switch fabrics described in this dissertation open up many new directions for research. One direction is the design of switch fabrics for ultra high throughput

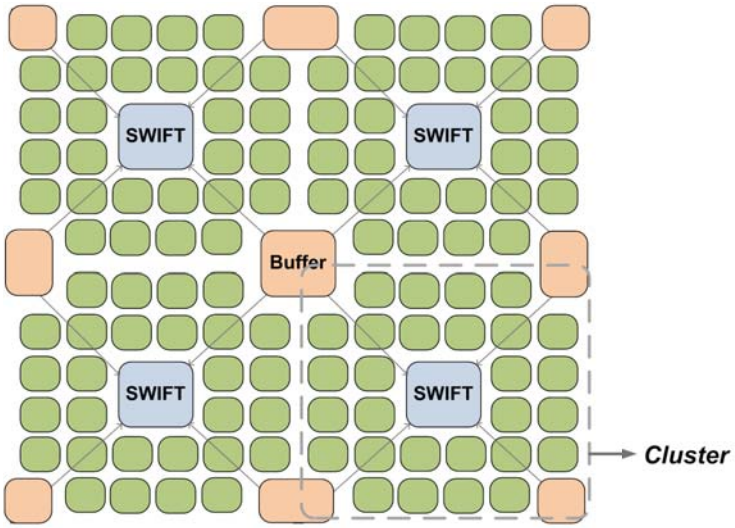

Figure 6.1: Ultra high radix switch fabric topology

exascale computing systems as shown in Fig. 6.1. The high radix switches proposed here can be used as basic building blocks to build ultra high radix communication platforms for such systems. Another area of exploration is 3D switch fabrics. The highly regular and modular architectures of SSN like switch fabrics make them very amenable for 3D integration. Through silicon vias (TSVs) in modern day 3D processes span only tens of micrometers and hence make them excellent candidate to replace bit-lines in our proposed switch fabrics as shown in Fig. 6.2. By leveraging 3D technology and the circuit techniques proposed in this dissertation, switch fabrics with bandwidth exceeding tens of Tb/s can be realized at very low latency. This will not only facilitate high bandwidth inter-core communication but also communication with stacked DRAM which currently limit the computation capabilities of most multi-core systems.

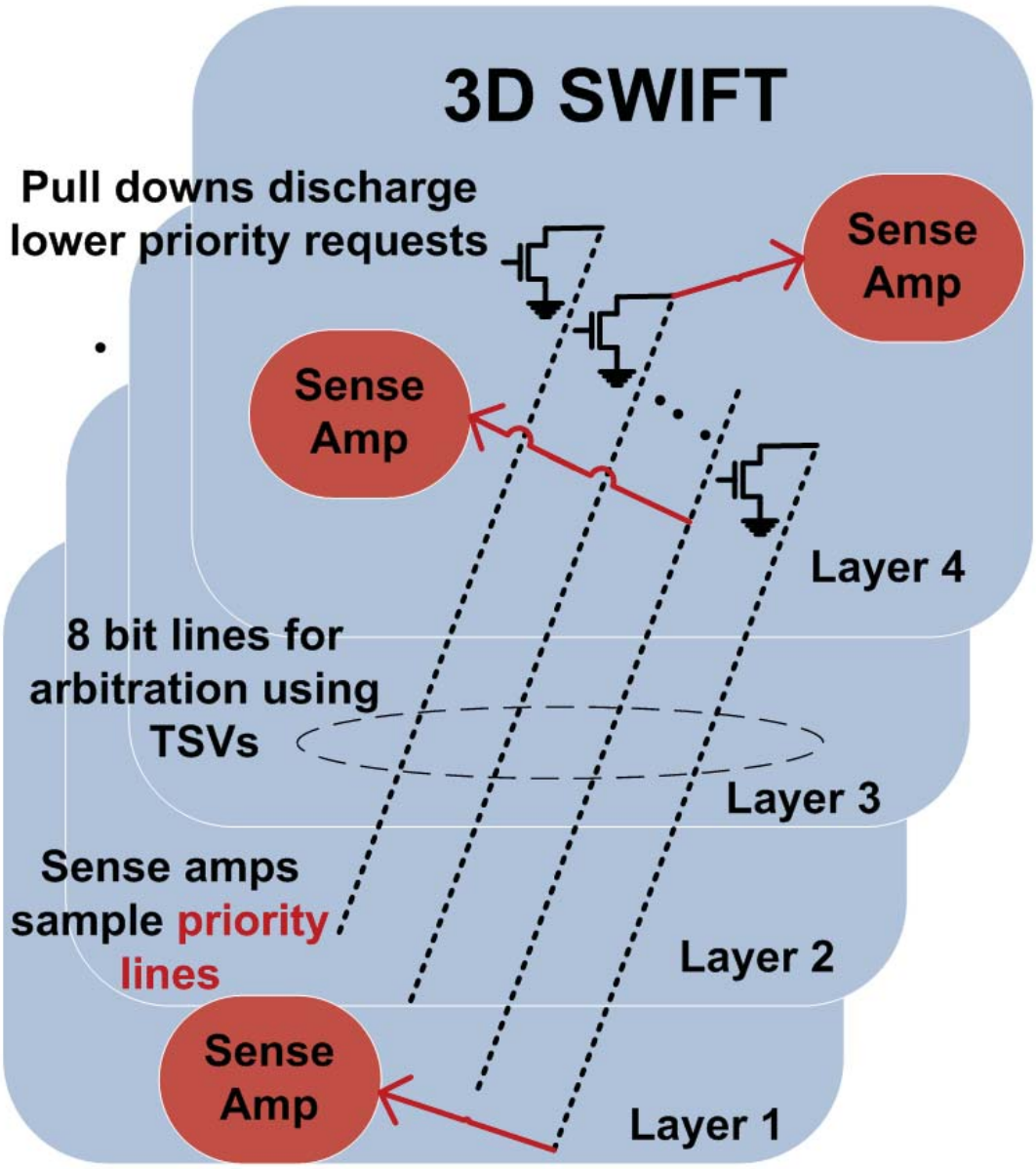

Figure 6.2: 3D SWIFT topology

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] "*The technical impact of Moores Law*", IEEE Solid-State Circuits Society Newsletter, Vol. 20, No. 3, Sep 2006.

[2] "*ISSCC 2011 trends report*", www.isscc.org/doc/2011/2011_Trends.pdf.

[3] J.Warnock et al., "*A 5.2Ghz Microprocessor Chip for the IBM zEnterprise^{TM} System*", IEEE International Solid State Circuits Conference, 2011.

[4] S.Sawant et al., "*A 32nm Westmere-EX Xeon Enterprise Processor*", IEEE International Solid State Circuits Conference, 2011.

[5] W.Hu et al., "*Godson-3B: A 1Ghz 40w 8-core 128GFlops processor in 65nm CMOS*", IEEE International Solid State Circuits Conference, 2011.

[6] S.Ruepp et al., "*Evaluation of 100 Gigabit Ethernet Switches under Bursty Traffic*", International Conference on Optical Network Design and Modelling, pp. 1-6, 2011.

[7] H-E.Kim et al., "*A 275mw heterogeneous Multimedia processor for IC-Stacking on Si-interpose*", IEEE International Solid State Circuits Conference, 2011.

[8] P-K.Tsung et al., "*A 216fps 4096×2160p 3dTv Set-Top Box Soc for Free-viewpoint 3DTV applications*", IEEE International Solid State Circuits Conference, 2011.

[9] E.Karl et al., "*ElastIC: An Adaptive Self-Healing Architecture for Unpredictable Silicon*", IEEE Design and Test of Computers , pp. 484-490, 2006.

[10] S.Vangal et al., "*An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS*", IEEE International Solid State Circuits Conference, pp. 98-99, 2007.

[11] M.Taylor et al., "*The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs*", IEEE Micro, 2002.

[12] J. Howard et al., "*A 48-Core IA-32 Processor in 45nm CMOS using On-Die Message-Passing and DVFS for Performance and Power scaling*", IEEE Journal of Solid State Circuits, Vol. 46, No. 1, Jan 2011.

[13] D. Truong et al., "*A 167-Processor Computational Platform in 65nm CMOS*", IEEE Journal of Solid State Circuits, pp. 1130-1144, April 2009.

[14] Y. Hung et al., "*Parallel Implementation and Performance Prediction of Object Detection in videos on the Tilera Many-Core Systems*", International Symposium on Pervasive Systems, Algorithms, and Networks, pp. 563-567, 2009.

[15] G. Passas et al., "*A 28×128×24Gb/s Crossbar Interconnecting 128 Tiles in a Single Hop and Occupying 6% of their area*", International Symposium on Networks-on-Chip, pp. 87-95, 2010.

[16] S. Murali et al., "*An Application-Specific Design Methodology for STbus Crossbar Generation*", Design Automation and Test in Europe, pp. 87-95, 2005.

[17] D.Flynni et al., "*AMBA: enabling reusable on-chip designs*", IEEE Micro, pp. 20-27, 1997.

[18] "*AMBA AXI Specification*", http://www.arm.com/armtech/AXI.

[19] "*AHB CLI Specification*", http://www.arm.com/armtech/ahbcli.

[20] P. Kongetira et al., "*Niagara: A 32-way multithreaded Sparc processor*", IEEE Micro, pp. 21-29, 2005.

[21] F. Wang et al., "*Fast fair arbiter design in packet switches*", Workshop on High Performance Switching and Routing, pp. 472-476, 2005.

[22] E. Shin et al., "*Round-robin Arbiter Design and Generation*", International Symposium on System Synthesis, pp. 243-248, 2002.

[23] Z. Yun et al., "*RR-LQD: A novel scheduling algorithm for CICQ switching fabrics*", International Symposium on System Synthesis, pp. 243-248, 2002.

[24] P. Gupta et al., "*Design and implementing a fast crossbar scheduler*", IEEE Micro, pp. 20-28, 1999.

[25] Z. Dong et al., "*Non-blocking memory-memory-memory Clos-network packet switch*", IEEE Sarnoff Symposium, pp. 1-5, 2011.

[26] A. Bouhraoua et al., "*A simplified router architecture for the modified FAT Tree Network-on-Chip Topology*", NORCHIP , pp. 1-4, 2009.

[27] A. Lea et al., "*The Load-Sharing Banyan Network*", IEEE Transactions on Computers, pp. 1025-1034, 2006.

[28] J. Kim et al., "*Flattened Butterfly Topology for On-Chip Networks*", IEEE Computer Architecture Letters, pp. 37-40, 2007.

[29] S. Bourduas et al., "*A Hybrid Ring/Mesh Interconnect for Network-on-Chip Using Hierarchical Rings for Global Routing*", International Symposium on Newtorks-on-Chip, pp. 195-204, 2007.

[30] C. Wang et al., "*A 1.1 GOPS/mW FPGA Chip with Hierarchical Interconnect Fabric*", IEEE International Symposium on VLSI Circuits, pp. 136-137, 2011.

[31] T. Krishna et al., "*NOCHI: Network-on-Chip with Hybrid Interconnect*", IEEE Micro, 2009.

[32] J. Frias et al., "*A VLSI crossbar switch with wrapped wave front arbitration*", IEEE Transactions on Circuits and Systems, pp. 135-141, 2003.

[33] J. Calvo et al., "*Asynchronous Modular Arbiter*", IEEE Transactions on Computers, pp. 67-70, 1986.

[34] S. Mahmud et al., "*A new arbitration circuit for asynchronous multiple bus multiprocessor systems*", IEEE International Symposium on Circuits and Systems, pp. 1041-1044, 1991.

[35] S. Zheng et al., "*Algorithm-Hardware Codesign of Fast Parallel Round-Robin Arbiters*", IEEE Transactions on Parallel and Distributed Systems, pp. 84-95, 2007.

[36] Y. Lee et al., "*A high-speed decentralized arbiter design for NoC*", IEEE International Conference on Computer Systems and Applications, pp. 350-353, 2009.

[37] K. Kim et al., "*A 125 GOPS 583 mW Network-on-Chip BasedParallel Processor With Bio-Inspired Visual Attention Engine*", IEEE Journal of Solid State Circuits, vol. 44 pp. 133-147, 2009.

[38] V. Shurbanov et al., "*The Effect of the Router Arbitration Policy on the Scalability of ServerNet^{TM} Topologies*", IEEE Symposium on Parallel and Distributed Processing, pp. 604-609, 1998.

[39] T. Seceleanu et al., "*Starvation-Free Arbitration Policies for the Segmented-Bus Platform*", IEEE Symposium on Signals, Circuits and Systems, pp. 67-70, 2005.

[40] P.J Garcia et al., "*Evaluation of Alternative Arbitration Policies for Myrinet Switches*", .

[41] M. Pirvu et al., "*The Impact of Link Arbitration on Switch Performance*", Proceedings of the 5^{th} International Symposium on High Performance Computer Architecture, 1999..

[42] S. Satpathy et al., "*A 4.5Tb/s 3.4Tb/s/W* 64×64 *switch fabric with self-updating least recently granted priority and quality of service arbitration in 45nm CMOS*", International Solid State Circuits Conference, 2012.

[43] K. Chang et al., "*A 50Gb/s 32×32 CMOS Crossbar chip using asymmetric serial links*", , IEEE International Symposium on VLSI Circuits, pp. 19-22, 1999.

[44] K. Goossens et al., "*Internet-Router Buffered Crossbars based on Networks on Chip*", Euromicro Conference on Digital System Design, Architectures, Methods, and Tools , pp. 365-374, 2009.

[45] B. Neji et al., "*Multistage Interconnection Network for MPSoC: Performance study and prototyping on FPGA*", International Design and Test workshop, pp. 11-16, 2008.

[46] P. Salihundam et al., "*A 2Tb/s 6/times4 Mesh Network with DVFS and 2.3Tb/s/W router in 45nm CMOS*", International Symposium on VLSI Circuits, pp. 79-80, 2010.

[47] S.Bell et al., "*Tile64 Processor: A 64-Core SoC with Mesh interconnect*", International Solid State Circuits Conference, pp. 88-89, 2008.

[48] M. Borgatti et al., "*A multi-context 6.4Gb/s/channel on-chip communication*

network using 0.18μm Flash-EEPROM switches and elastic interconnects", International Solid State Circuits Conference, pp. 466-467, 2003.

[49] S. Rodrigo et al., "*Efficient unicast and multicast support for CMPs*", IEEE MICRO, pp. 364-375, 2008.

[50] S. Satpathy et al., "*A 1.07 Tb/s 128×128 Swizzle network for SIMD Processors*", International Symposium on VLSI Circuits, pp. 81-82, 2010.

[51] S. Satpathy et al., "*SWIFT: A 2.1Tb/s* 32×32 *Self-Arbitrating Manycore Interconnect Fabric*", International Symposium on VLSI Circuits, pp. 138-139, 2010.

[52] M. Woh et al., "*Low power interconnects for SIMD computers*", Design, Automation and Test in Europe Conference, pp. 1-6, 2011.

[53] M. Woh et al., "*AnySP: Anytime Anywhere Anyway Signal Processing, International Symposium on Microacrhitecture*", International Symposium on Microacrhitecture, pp. 81-91, 2010.

[54] M. Woh et al., "*Analyzing the scalability of SIMD for the next generation software defined radio*", ICASSP pp. 5388-5391, 2008.

[55] Y. Lin et al., "*SODA: A low-power architecture for software radio*", International Symposium on Computer Architecture, pp. 89-101, 2006.

[56] S. Bell et al., "*Tile64 Processor: A 64-Core SoC with Mesh Interconnnect*", ISSCC Dig. Tech. Papers, pp. 88-89, 2008.

[57] S. Tremblay et al., "*A Third-Generation 65nm 16-Core 32-Thread Plus 32-Scout-Thread CMT SPARC Processor*", ISSCC Dig. Tech. Papers, pp. 82-83, 2008.

[58] M. Anders et al., "*A 4.1Tb/s Bisection-Bandwidth 560Gb/s/W Streaming Circuit-Switched 8×8 Mesh Network-on-Chip in 45nm CMOS*", ISSCC Dig. Tech. Papers, pp. 110-111, 2010.

[59] S. Vangal et al., "*A 5.1GHz 0.34mm^2 Router for Network-on-Chip Applications*", International Symposium on VLSI Circuits, pp. 42-43, 2007.

[60] M. Lee et al., "*Probabilistic Distance-based Arbitration: Providing Equality of Service for Many-core CMPs*", IEEE MICRO43, 2010.

[61] C. Park et al., "*A 1.2 TB/s on-chip ring interconnect for 45nm 8-core enterprise Xeon processor*", ISSCC Dig. Tech. Papers, pp. 180-181, 2010.

[62] B.Stackhouse et al., "*A 65nm 2-Billion Transistor Quad-Core Itanium Processor*", JSSCC pp. 18-31, Vol. 44, No. 1, January 2009.

[63] B.Kim et al., "*A 4Gb/s/ch 356fJ/b 10mm Equalized On-chip Interconnect with Nonlinear Charge-Injecting Transmit Filter and Transimpedance Receiver in 90nm CMOS*", ISSCC Dig. Tech. Papers, pp. 66-67, 2009.

[64] J.Seo et al., "*High Bandwidth and Low Energy On-Chip Signaling with Adaptive Pre-Emphasis in 90nm CMOS*", ISSCC Dig. Tech. Papers, pp. 182-183, 2010.

[65] R. Ho et al., "*High-Speed and Low-Energy Capacitively-Driven On-Chip Wires*", ISSCC Dig. Tech. Papers, pp. 412-413, 2007.

[66] E. Mensink et al., "*A 0.28pJ/b 2Gb/s/ch Transceiver in 90nm CMOS for 10mm On-chip interconnects*", ISSCC Dig. Tech. Papers, pp. 414-415, 2007.